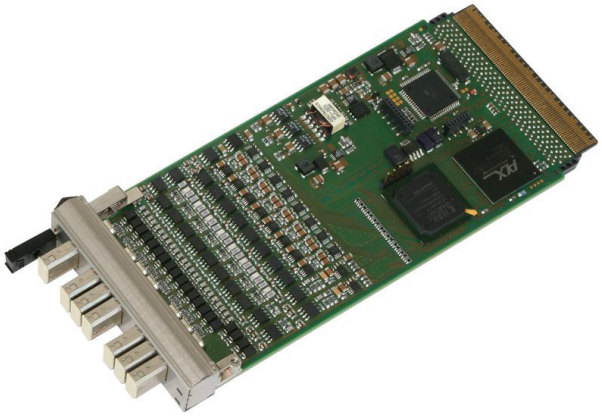
AMC-ADIO24
AMC-ADIO24-HD50

AMC Analog/Digital I/O Module



Hardware Manual

to Products U.1001.01,
U.1001.02,
U.1001.09,
U.1001.10

powerBridge Computer

Ehlbeek 15a
30938 Burgwedel
fon 05139-9980-0 www.powerbridge.de
fax 05139-9980-49 info@powerbridge.de

esd electronic system design gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
http://www.esd.eu
Phone: +49 (0) 511 3 72 98-0 • Fax: +49 (0) 511 3 72 98-68

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. In particular descriptions and technical data specified in this document may not be constituted to be guaranteed product features in any legal sense.

esd reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone:	+49-511-372 98-0
Fax:	+49-511-372 98-68
E-Mail:	info@esd.eu
Internet:	www.esd.eu

⚠	This manual contains important information and instructions on safe and efficient handling of the AMC-ADIO24/-HD50. Carefully read this manual before commencing any work and follow the instructions. The manual is a product component, please retain it for future use.

Document file:	I:\Texte\Doku\MANUALS\AMC\AMC-ADIO24\Englisch\AMC-ADIO24_Hardware_en_21.odt
Date of print:	2018-01-09
Document type number:	DOC0800

Hardware version:	AMC-ADIO: Rev. 2.01 AMC-ADIO24-HD50: Rev.1.0 AMC-ADIO-Adapter: Rev. 1.0 from serial number GB000024 and higher AMC-ADIO24-HD50-Adapter Rev. 1.0
Firmware version:	from FPGA release 2.0 on

Document History

The changes in the document listed below affect changes in the hardware as well as changes in the description of the facts, only.

Revision	Chapter	Changes versus previous version	Date
1.0	-	First released version of English manual.	2011-01-10
2.0	-	Description of AMC-ADIO24-HD50 inserted	2013-01-30
	-	Safety instructions revised, note to shielded cables inserted	
	-	Hardware version of AMC-ADIO-Adapter inserted (serial number GB000024 and higher)	
	1.	Description of AMC-ADIO24-HD50 inserted, block circuit diagram new	
	2.2	PCB view of AMC-ADIO24-HD50 inserted	
	3.	Description of AMC-ADIO24-HD50 inserted	
	3.1.1	AMC-LED1 and LED2 changed in figure	
	3.1.2	Front panel LED and connector position of AMC-ADIO24-HD50 inserted	
	4.1.2	Description of PEX 8311	
	4.2	Note inserted	
	4.2.1	Chapter inserted:"esdmio driver user-level API"	
	4.6, 4.7.23	Description of DOWrite A/B changed	
	4.7	Description of format of analog data inserted	
	4.7.5	Content of register DIOMode inserted in table	
	4.7.22, 4.7.27.1, 4.7.28.1	Heading corrected ADC0 ... ADC7	
	4.7.25	Format and description of ChanWrite corrected (rrr), note inserted	
	4.8	Figure corrected	
	4.9.1.2	Correction in 1. DMA-enabled: WA-RA>0	
	5.	Technical data of AMC-ADIO24-HD50 inserted	
	6.	Chapter revised, connector assignment of AMC-ADIO24-HD50 new	
	6.6, 6.7	Description of 50-pin har.mik connector inserted	
	7.	Figure 8 and 10 changed	
	9.	Declaration of conformity inserted	
2.1	-	Classification of safety instructions inserted	2018-01-09
	5.1	Description of dimension corrected	
	6.	Chapter restructured	
	7.	Chapter “AMC-ADIO-Adapter for AMC-ADIO24 ” moved	
	8.	Chapter “Connector Assignment of AMC-ADIO24-HD50” moved	
	9.	Chapter “AMC-ADIO24-HD50-Adapter ” new	
	11.	Declaration of Conformity new	
	12.	Product AMC-ADIO24-HD50-Adapter inserted	

Technical details are subject to change without further notice.

Classification of Warning Messages and Safety Instructions

This manual contains noticeable descriptions, warning messages and safety instructions, which you must follow to avoid personal injuries or death and property damage.

This is the safety alert symbol.
It is used to alert you to potential personal injury hazards. Obey all safety messages and instructions that follow this symbol to avoid possible injury or death.

DANGER, WARNING, CAUTION

Depending on the hazard level the signal words DANGER, WARNING or CAUTION are used to highlight safety instructions and warning messages. These messages may also include a warning relating to property damage.

DANGER
Danger statements indicate a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING
Warning statements indicate a hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION
Caution statements indicate a hazardous situation that, if not avoided, could result in minor or moderate injury.

NOTICE

Notice statements are used to notify people on hazards that could result in things other than personal injury, like property damage.

NOTICE
This NOTICE statement indicates that the device contains components sensitive to electrostatic discharge.

NOTICE
This NOTICE statement contains the general mandatory sign and gives information that must be heeded and complied with for a safe use.

INFORMATION

INFORMATION
Notes to point out something important or useful.

Safety Instructions

- When working with the AMC-ADIO24/-HD50 follow the instructions below and read the manual carefully to protect yourself from injury and the AMC-ADIO24/-HD50 from damage.
- The device is a built-in component. It is essential to ensure that the device is mounted in a way that cannot lead to endangering or injury of persons or damage to objects.
- Do not use damaged or defective cables to connect the AMC-ADIO24/-HD50.
- In case of damages to the device, which might affect safety, appropriate and immediate measures must be taken, that exclude an endangerment of persons and domestic animals and property.
- Current circuits which are connected to the device have to be sufficiently protected against hazardous voltage (SELV according to EN 60950-1).
- The AMC-ADIO24/-HD50 may only be driven by power supply current circuits, that are contact protected.
  A power supply, that provides a safety extra-low voltage (SELV) according to EN 60950-1, complies with this conditions.

- Do not open the housing of the AMC-ADIO24/-HD50-Adapter.
- The device has to be securely installed in the control cabinet before commissioning.
- Protect the AMC-ADIO24/-HD50 from dust, moisture and steam.
- Protect the AMC-ADIO24/-HD50 from shocks and vibrations.
- The AMC-ADIO24/-HD50 may become warm during normal use. Always allow adequate ventilation around the AMC-ADIO24/-HD50 and use care when handling.
- Do not operate the AMC-ADIO24/-HD50 adjacent to heat sources and do not expose it to unnecessary thermal radiation. Ensure an ambient temperature as specified in the technical data.

NOTICE
Electrostatic discharges may cause damage to electronic components.
To avoid this, perform the steps described on page 13 *before* you touch the AMC-ADIO24/-HD50, in order to discharge the static electricity from your body.

Qualified Personnel
This documentation is directed exclusively towards personnel qualified in control and automation engineering.
The installation and commissioning of the product may only be carried out by qualified personnel, which is authorized to put devices, systems and electric circuits into operation according to the applicable national standards of safety engineering.

Conformity
The AMC-ADIO24 is an industrial product and meets the demands of the EU regulations and EMC standards printed in the conformity declaration at the end of this manual.

Warning: In a residential, commercial or light industrial environment the AMC-ADIO24/-HD50 may cause radio interferences in which case the user may be required to take adequate measures.

NOTICE
To sustain the compliance with directive 2014/30/EC, it is necessary to use shielded cables.

Intended Use
The intended use of the AMC-ADIO24/-HD50 is the operation as AMC analog/digital IO module in a µTCA system.
The guarantee given by esd does not cover damages which result from improper use, usage not in accordance with regulations or disregard of safety instructions and warnings.

- The AMC-ADIO24/-HD50 is intended for installation in a MicroTCA-system only.
- The operation of the AMC-ADIO24/-HD50 in hazardous areas, or areas exposed to potentially explosive materials is not permitted.
- The operation of the AMC-ADIO24/-HD50 for medical purposes is prohibited.

Service Note
The AMC-ADIO24/-HD50 does not contain any parts that require maintenance by the user. The AMC-ADIO24/-HD50 does not require any manual configuration of the hardware.

Disposal
Devices which have become defective in the long run have to be disposed in an appropriate way or have to be returned to the manufacturer for proper disposal. Please make a contribution to environmental protection.

Number Representation
All numbers in this document are base 10 unless designated otherwise. Hexadecimal numbers have a prefix of 0x. For example, 42 is represented as 0x2A in hexadecimal.

Table of contents

Safety Instructions....5

1. Overview....10

2. PCB View with Connectors....11
2.1 AMC-ADIO24 (U.1001.01)....11
2.2 AMC-ADIO24-HD50 (U.1001.02)....12

3. Hardware Installation....13
3.1 Front Panel LED and Connector Positions....14
3.1.1 AMC-ADIO24....14
3.1.2 AMC-ADIO24-HD50....15
3.1.3 LED Indication....15

4. PCI Express Device Access....16
4.1 PCI Address Space....16
4.2 Software Development Kits for the PCI Bridge PEX8311....18
4.2.1 esdmio Driver user-level API....18
4.2.1.1 esdmio_DeviceOpen....19
4.2.1.2 esdmio_DeviceClose....19
4.2.1.3 esdmio_DeviceInfo....20
4.2.1.4 esdmio_MapRegisters....21
4.2.1.5 esdmio_MapDMA....21
4.2.1.6 esdmio_DMASingle....22
4.2.1.7 esdmio_DMAMultiStart....23
4.2.1.8 esdmio_DMAMultiStop....24
4.2.1.9 esdmio_DMAPoll....24
4.2.1.10 esdmio_DMAWait....25
4.2.1.11 IRQWait....25
4.2.1.12 Return Codes....26
4.2.1.13 Example Source....27
4.3 Data Access....30
4.3.1 Direct I/O Access....30
4.3.2 Setting Output Data....31
4.3.3 Periodic Pattern Generation....31
4.4 Streaming I/O and Trigger Selection....32
4.4.1 Streaming I/O Block Diagram....32
4.4.2 Trigger Selection....33
4.4.2.1 Overview....33
4.4.2.2 Timing Routing Pool....33
4.5 Store/Restore Default Parameters (Calibration, FRU, FPGA-Image)....34
4.6 FPGA Memory Map....35
4.7 Register Description....38
4.7.1 Version....38
4.7.2 Status....39
4.7.3 DDI (Direct Digital In, Inclusive Inversion)....41
4.7.4 DDO (Direct Digital Out, No Inversion)....42
4.7.5 DIOMode (Trigger Mode Din/Dout)....43
4.7.6 DOut Invert (Digital Output Inverted)....44
4.7.7 HSE/LSE (HighSide Enable / LowSide Enable)....45
4.7.8 FGENAB (DDFS Frequency Generator)....46
4.7.9 FGENCD (Clock Divider)....47
4.7.10 SyncOut (External Trigger Output/Input)....48
4.7.11 DORout (Trigger Output/Input, DigOut-Route)....49
4.7.12 DInInvert (Digital Input Inverted)....50
4.7.13 DInFilter (Filter Digital Input)....51

4.7.14 ADCMode (Trigger Mode ADC7...0)....52
4.7.15 DIVMode (DMA-Mode, Timer Select, Interrupt Enable)....53
4.7.16 Auxreg (Trigger Mode DAC A/B)....56
4.7.17 QDac (Direct Read Back DAC A/B)....57
4.7.18 DACCorr (Gain/Offset Correction DAC B/A)....58
4.7.19 EVENT....59
4.7.20 DMAMode (DMA enable/disable)....61
4.7.21 Compare....63
4.7.22 ADC0Corr ... ADC7Corr (ADC Gain and Offset)....64
4.7.23 DOWriteX (Digital Out Mask)....65
4.7.24 DACWriteBA (Write DAC)....66
4.7.25 ChanWrite....67
4.7.26 Current Register Values....68
4.7.26.1 ADCxAct (Current Analog Input Values)....68
4.7.26.2 DInAAct, DInBAct (Current Digital Input Values)....69
4.7.26.3 DOutAAct, DOutBAct (Current Digital Output Values)....69
4.7.26.4 DACAAct, DACBAct (Current Analog Output Values)....70
4.7.26.5 TSAct (Current Timestamp Counter Value)....71
4.7.27 Last DMA Values....72
4.7.27.1 ADC0Last... ADC7Last (Last Analog Input DMA Values)....72
4.7.27.2 DInALast, DInBLast (Last Digital Input DMA Values)....73
4.7.27.3 DOutALast, DOutBLast (Last Digital Output DMA Values)....73
4.7.27.4 DACALast, DACALast (Last Analog Output DMA Values)....74
4.7.27.5 TSLast (Last TS-Counter DMA Values)....74
4.7.28 Difference Values for Trigger Conditions 75
4.7.28.1 ADC0Delta … ADC7Delta....75
4.7.28.2 DInA/BMask....76
4.7.28.3 DOutA/BMask....77
4.7.28.4 DACA/BDelta....78
4.7.28.5 TSDelta....79
4.7.29 Command Register....80
4.8 I/O Data Processing....82
4.9 DMA Access....83
4.9.1 DMA Input Data....83
4.9.1.1 Read Data FIFO....83
4.9.1.2 Setting DMA Operation....85
4.9.2 DMA Output Data....86

5. Technical Data....87
5.1 General Technical Data....87
5.2 MicroTCATM /AMC Standards....88
5.3 Analog Outputs....88
5.4 Analog Inputs....89
5.5 Digital Inputs/Outputs....90
5.6 Trigger Ports....90

6. Connector Assignment of AMC-ADIO24....91
6.1 Analog Out....91
6.2 Analog In....92
6.3 Digital I/O....93
6.4 Trigger Ports....94

7. AMC-ADIO-Adapter for AMC-ADIO24 95
7.1 AMC-ADIO-Adapter View....96
7.2 Technical Data AMC-ADIO-Adapter....96
7.3 Connector Assignment....97

8. Connector Assignment of AMC-ADIO24-HD50....99
8.1 Trigger Ports....99

8.2 ADIO via 50-pin har.mik connector 100

9. AMC-ADIO24-HD50-Adapter 101
9.1 Technical Data AMC-ADIO24-HD50-Adapter 102
9.2 Connector Assignment 103

10. Abbreviations 104

11. Declaration of Conformity of AMC-ADIO24 105

12. Order Information 106

1. Overview

INFORMATION
The AMC-ADIO24 and the AMC-ADIO24-HD are identically constructed except of the har-link®/har.mik® connectors. Furthermore the functionalities are identical. In this manual both modules are described together (AMC-ADIO24/-HD50), differences are specified in the corresponding sections.

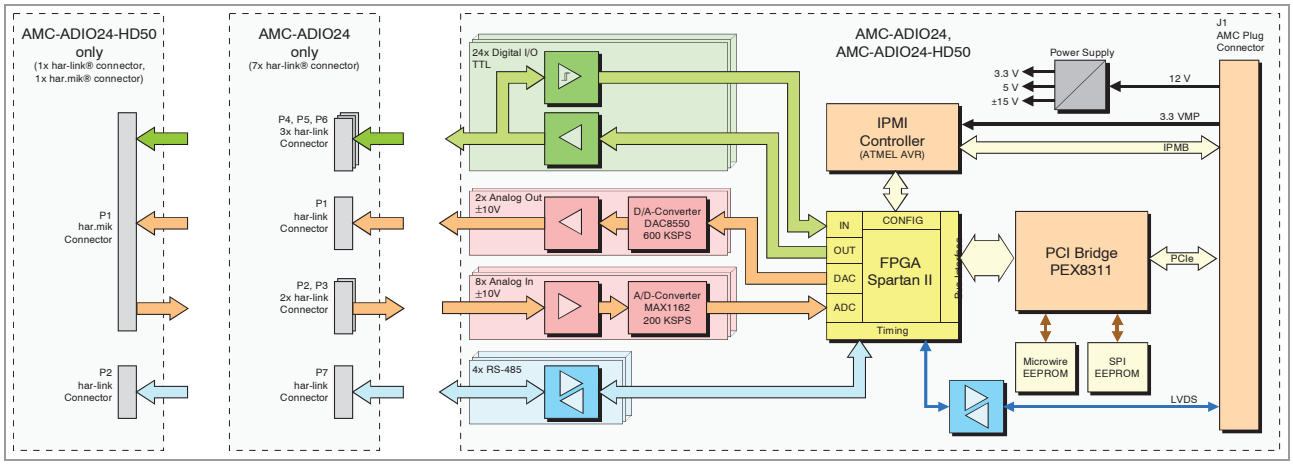


Figure 1: Block circuit diagram

The AMC-ADIO24/-HD50 is an AMC analog/digital I/O module.
It is equipped with a Spartan FPGA, which manages the I/O data exchange in cooperation with the PCIe bridge. FIFOs for input and output direction and DMA to the PCIe host CPU's memory minimizes undesired latency during PCIe read cycles at higher data rates. Read cycles of the PCIe CPU are reduced to set-up and diagnosis tasks.

The AMC-ADIO24 (esd order No.: U.1001.01) is equipped with 7 Harting® har-link® connectors.
The AMC-ADIO24-HD50 (esd order-No.: U.1001.02) is equipped with one Harting® har.mik® connector and one Harting® har-link® connector.

The eight overvoltage protected analog inputs are connected to eight 16-bit A/D converters with a sampling rate of up to 200 kHz each.
Both 16 bit analog outputs have a differential and a single ended output circuit, accessible at separate pins at the connector. The outputs are transient and short circuit protected. The maximum update rate of the analog outputs is 600 KSPS (kilo samples per second).
Each of the 24 TTL-level I/Os can be separately configured as input or output. The outputs can be configured as high side driver, low side driver or both (sink/source). The I/O port state can be read back in any configuration via a comparator with hysteresis.

For the trigger I/Os for synchronisation the firmware offers a so called 'Timing-Routing-Pool' with various trigger conditions (RS-485 trigger input, timer, software, free run) that can be individually evaluated for each I/O or I/O group.
For setup and I/O data exchange a comprehensive register description is accessible via PCIe (see chapter 4).

2. PCB View with Connectors

2.1 AMC-ADIO24 (U.1001.01)

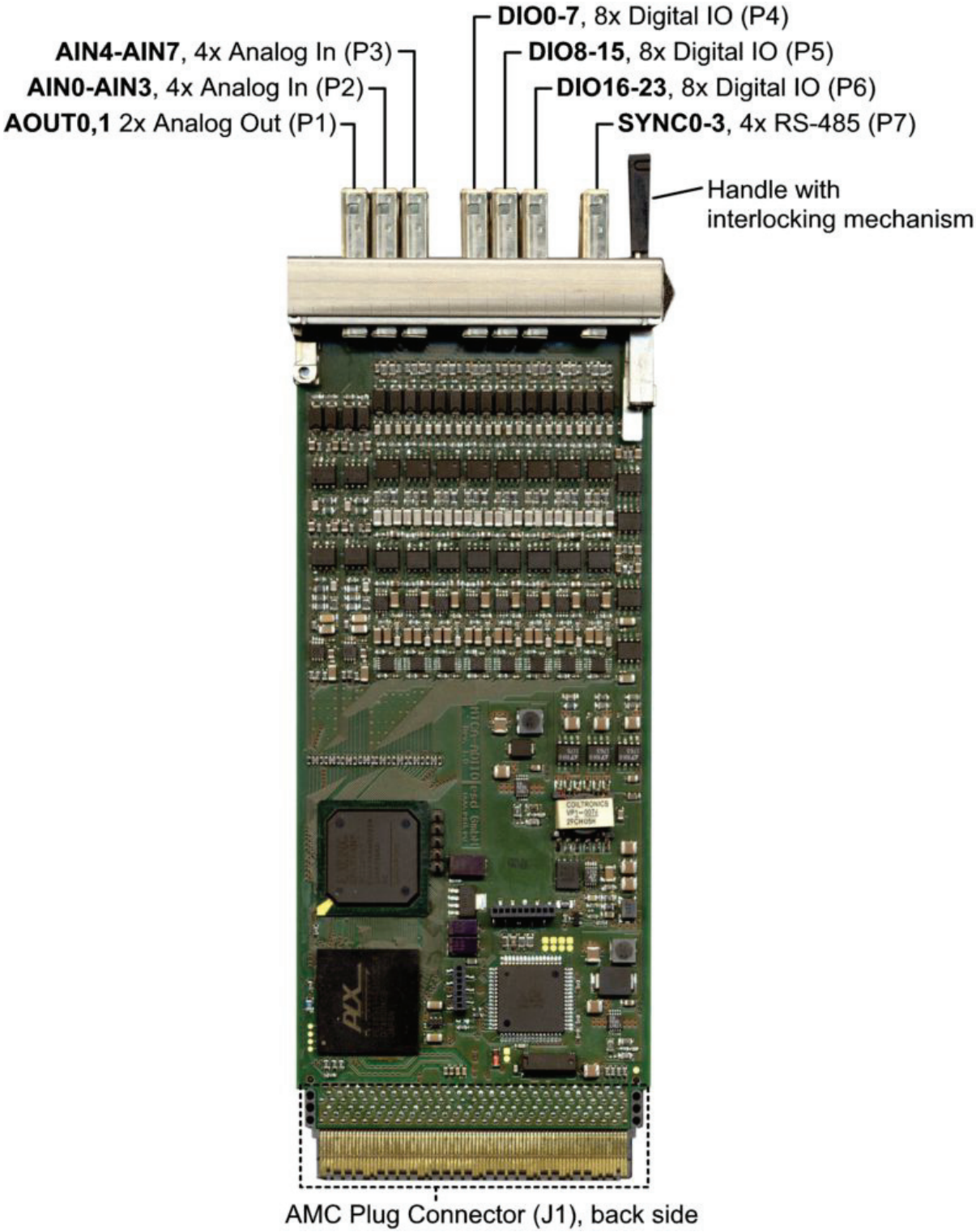


Figure 2: PCB top view of AMC-ADIO24

See also page 91 et seqq. for signal assignments of the connectors.

2.2 AMC-ADIO24-HD50 (U.1001.02)

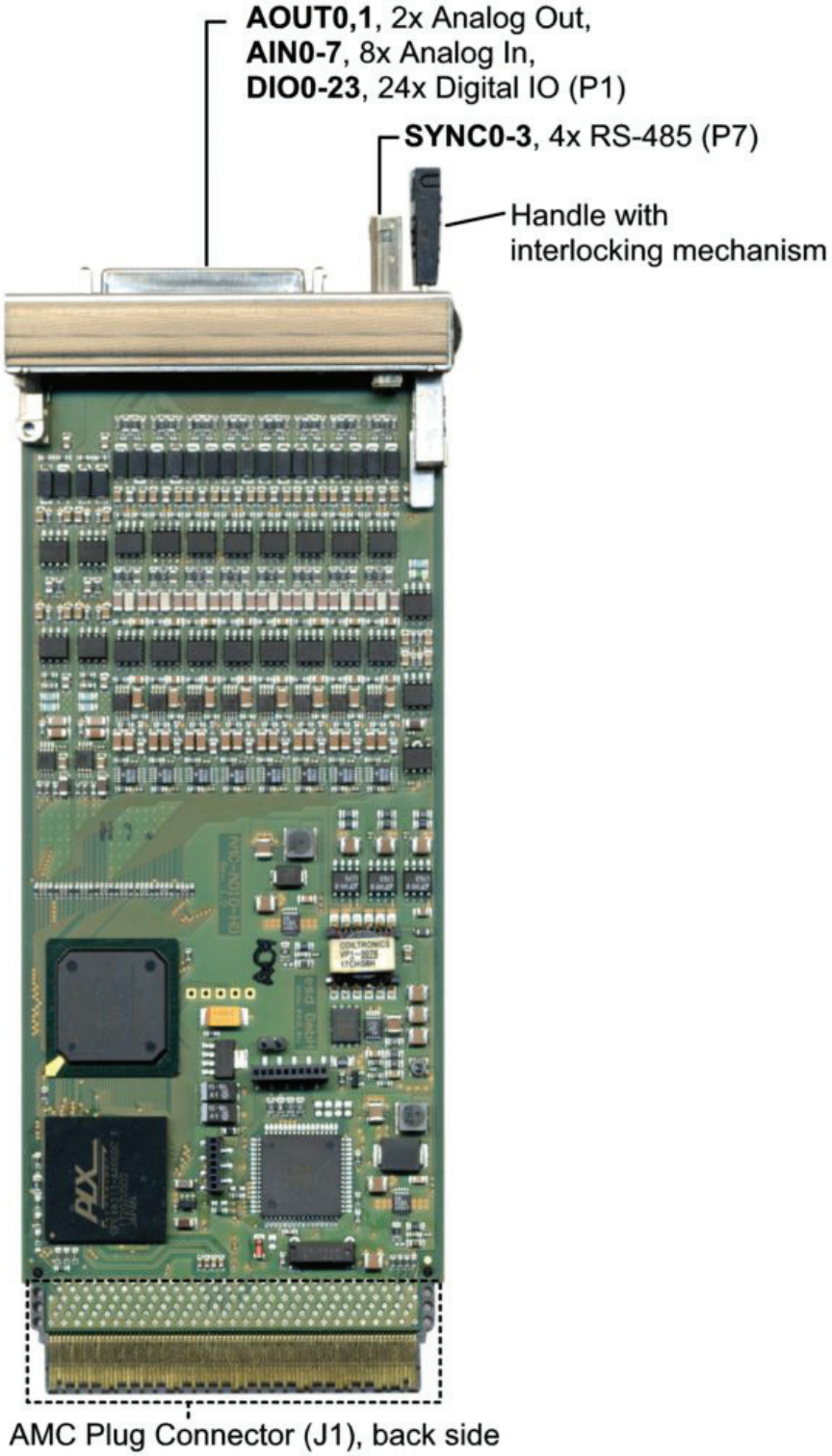


Figure 3: PCB top view of AMC-ADIO24

See also page 99 et seqq. for signal assignments of the connectors.

3. Hardware Installation

NOTICE
Read the safety instructions at the beginning of this document carefully, before you start with the hardware installation!

NOTICE
Electrostatic discharges may cause damage to electronic components.

→ To avoid this, please discharge the static electricity from your body by touching the metal case of the µTCA system *before* you touch the AMC-ADIO24/-HD50.
→ Furthermore, you should prevent your clothes from touching the AMC-ADIO24/-HD50, because your clothes might be electrostatically charged as well.

Procedure:

1. The AMC-ADIO24/-HD50 is hot-pluggable.
   Insert the AMC-ADIO24/-HD50 into a free slot in your µTCA system.
2. Fix the AMC-ADIO24/-HD50 by pushing the handle with interlocking mechanism (see figure 4 for AMC-ADIO24 or figure 5 for AMC-ADIO24-HD50).
3. AMC-ADIO24: Connect the signal lines to the har-link-connectors (P1 - P7) in the front panel as described in figure 4, e.g. via the har-link® cable (see page 106).

   AMC-ADIO24-HD50: Connect the signal lines to the har.mik connector (P1) and the har-link connector P2 in the front panel as described in figure 5.

   AMC-ADIO24/-HD50: The external signal-lines are 'hot-pluggable' and can be connected or disconnected at discretion.
4. End of hardware installation.
5. Set the interface properties in your operating system. For further information refer to the documentation of the operating system.

3.1 Front Panel LED and Connector Positions

3.1.1 AMC-ADIO24

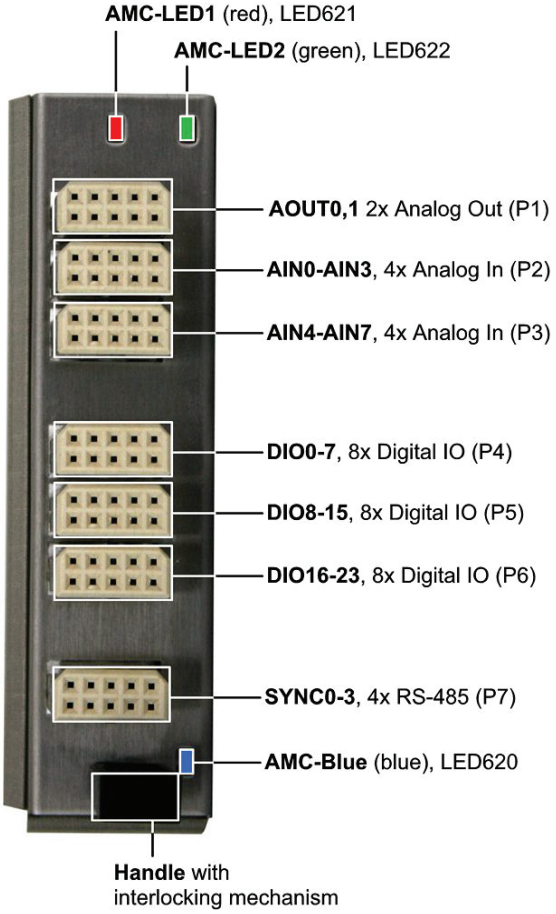


Figure 4: AMC-ADIO24 connectors and LEDs

3.1.2 AMC-ADIO24-HD50

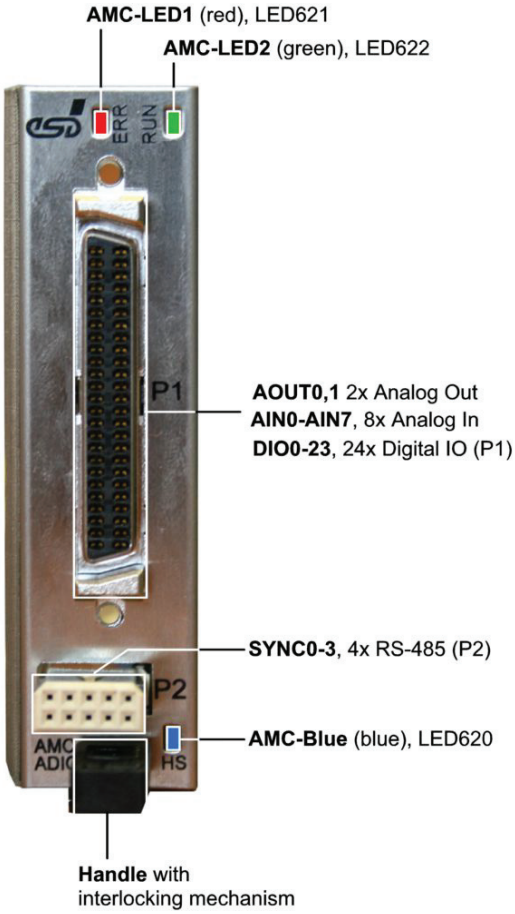


Figure 5: AMC-ADIO24-HD50 connectors and LEDs

3.1.3 LED Indication

Name	Colour	Description		LED name in schematics diagram
AMC-LED1	red	Controlled by IPMI.		LED621
AMC-LED2	green	Controlled by IPMI. Local function: Lit if FPGA is booted correctly.		LED622
AMC-Blue	blue	Controlled by IPMI.		LED620
		Off	in operation	
		Blinking	preparing for hot-plug (in transition)	
		Lit	powered off, hot-plug allowed	

Table 1: LED indication of AMC-ADIO24/-HD50

4. PCI Express Device Access

A 64K address space is required for the AMC-ADIO24/-HD50. The address space must be accessible by the host CPU of the PCIe bus.

4.1 PCI Address Space

The PCI base address of the AMC-ADIO24/-HD50 depends on the PCI host that assigns the PCI bus memory addresses to the connected PCI bus boards.

Note that the PCIe bridge of the AMC-ADIO24/-HD50 (PEX8311) internally carries two PCI-bridges, one 8111-compatible PCIe to PCI bridge and one 9056-compatible PCI to Local Bus (connected to the FPGA) bridge. Therefore the PCI host detects and displays two devices:

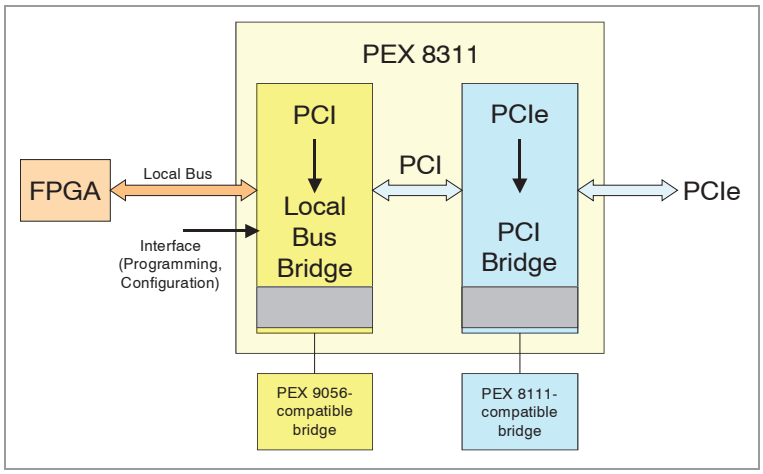


Figure 6: Internal bridges of PEX 8311

The following listing shows an example of the memory assignment at a x86 Linux system, after calling the command 'lspci -vv'.

```
…
...
06:00.0 PCI bridge: PLX Technology, Inc. PEX 8111 PCI Express-to-PCI Bridge (rev 21) (prog-if 00 [Normal decode])
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV+ VGASnoop- ParErr- Stepping- SERR- FastB2B-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Latency: 0, Cache Line Size: 32 bytes
        Region 0: Memory at fea00000 (64-bit, prefetchable) [size=64K]
        Bus: primary=06, secondary=07, subordinate=07, sec-latency=36
        I/O behind bridge: 00002000-00002fff
        Memory behind bridge: fe600000-fe6fffff
        Secondary status: 66MHz- FastB2B- ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort+ <SERR- <PERR-
        BridgeCtl: Parity- SERR- NoISA+ VGA- MAbort- >Reset- FastB2B-
        Capabilities: [50] Message Signalled Interrupts: Mask- 64bit+ Queue=0/0 Enable-
                Address: 0000000000000000  Data: 0000
        Capabilities: [60] Express PCI/PCI-X Bridge IRQ 0
                Device: Supported: MaxPayload 128 bytes, PhantFunc 0, ExtTag-
                Device: Latency L0s <64ns, L1 <1us
                Device: AtnBtn- AtnInd- PwrInd-
                Device: Errors: Correctable- Non-Fatal- Fatal- Unsupported-
                Device: RlxdOrd- ExtTag- PhantFunc- AuxPwr- NoSnoop-
                Device: MaxPayload 128 bytes, MaxReadReq 4096 bytes
                Link: Supported Speed 2.5Gb/s, Width x1, ASPM L0s L1, Port 0
                Link: Latency L0s <1us, L1 <16us
                Link: ASPM Disabled CommClk- ExtSynch-
                Link: Speed 2.5Gb/s, Width x1
        Capabilities: [100] Power Budgeting

07:04.0 Signal processing controller: PLX Technology, Inc. Francois (rev ba)
        Subsystem: ESD Electronic System Design GmbH Unknown device 0600
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV+ VGASnoop- ParErr- Stepping- SERR- FastB2B-
        Status: Cap+ 66MHz+ UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Latency: 32, Cache Line Size: 32 bytes
        Interrupt: pin A routed to IRQ 7
        Region 0: Memory at fe610000 (32-bit, non-prefetchable) [size=512]
        Region 1: I/O ports at 2000 [size=256]
        Region 2: Memory at fe600000 (32-bit, non-prefetchable) [size=64K]
        Capabilities: [40] Power Management version 2
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0-,D1-,D2-,D3hot-,D3cold-)
                Status: D0 PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [48] #06 [0000]
        Capabilities: [4c] Vital Product Data
…
...
```

For the user the memory base address of the **FPGA** at the AMC-ADIO24 is most important. It is always placed in the PCI bridge called

'Subsystem: ESD Electronic System Design GmbH...'

after the I/O-ports and is displayed as

'Region 2: Memory at fe600000 (32-bit, non-prefetchable) [size=64K]',

i.e. the base address is 0xFE600000 in this example.
The memory space size reserved for the FPGA registers always is defined as 64k.

4.2 Software Development Kits for the PCI Bridge PEX8311

For an easier start we recommend to use the esd driver for the AMC-ADIO24, called 'esdmio'. This kit contains an universal driver for Linux (with source code) that allows the access to the card (see chapter 4.2.1).
The Linux driver and library need to be built from source.

Another possibility is to use the SDK of PLX, called 'PlxSDK'. This kit contains an universal driver for Windows and Linux (with source code) that allows the access to the user memory space of the bridge.

The PlxSDK can be downloaded from the PLX homepage:
http://www.plxtech.com/products/sdk/pde

INFORMATION
The Plx9056 driver (and not the 8311 or Svc driver) is needed to get access to the FPGA.

4.2.1 esdmio Driver user-level API

In this chapter the functions of the user-level API of the esdmio library "libesdmio" and the return values are described. In addition, an example code is given.

Each function description is structured identically into a description of:

- Syntax
- Description
- Arguments
- Return Values
- Usage

4.2.1.1 esdmio_DeviceOpen

Syntax:

```
ESDMIO_RESULT esdmio_DeviceOpen
(
 int32_t         idx,
 ESDMIO_HANDLE   *hnd
);
```

Description: Opens a connection to an I/O card.

Arguments: *idx* is a number from 0..255 (actual number of supported cards by a driver may be less than 256)

hnd must point to an ESDMIO_HANDLE space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the `ESDMIO_HANDLE` is valid afterwards.

4.2.1.2 esdmio_DeviceClose

Syntax:

```
ESDMIO_RESULT esdmio_DeviceClose
(
 ESDMIO_HANDLE   *hnd
);
```

Description: Closes a connection to an I/O card.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the ESDMIO_HANDLE is invalid afterwards.

4.2.1.3 esdmio_DeviceInfo

Syntax:

```
ESDMIO_RESULT esdmio_DeviceInfo
(
 ESDMIO_HANDLE  hnd,
 ESDMIO_DEVINFO *info
);
```

Description: Returns information about an I/O card.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

info must point to an ESDMIO_DEVINFO space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the ESDMIO_DEVINFO contains valid data.

4.2.1.4 esdmio_MapRegisters

Syntax:

```
ESDMIO_RESULT esdmio_MapRegisters
(
 ESDMIO_HANDLE  hnd,
 uint32_t       **regs
);
```

Description: Maps the FPGA registers into user-space.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

regs must point to a pointer space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, **regs* will contain the address of the FPGA registers.

4.2.1.5 esdmio_MapDMA

Syntax:

```
ESDMIO_RESULT esdmio_MapDMA
(
 ESDMIO_HANDLE  hnd,
 uint32_t       **dma
);
```

Description: Maps the DMA memory into user-space.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

dma must point to a pointer space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, **dma* will contain the address of the DMA memory.

4.2.1.6 esdmio_DMASingle

Syntax:

```
ESDMIO_RESULT esdmio_DMASingle
(
 ESDMIO_HANDLE        hnd,
 ESDMIO_DMASETTINGS   *dmasetup
);
```

Description: Initiates a single (one-shot) DMA transfer of I/O data from the card to the DMA memory.

Arguments:

hnd must point to a valid ESDMIO_HANDLE.

dmasetup must point to a valid ESDMIO_DMASETTINGS.

For a single transfer, the parameter blocks must be 1.

The parameter blocksize must be a multiple of 16 and may not exceed the dma_size of the card.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the DMA transfer has been started and is running in background.

Usage: After starting a DMA transfer, one must either use *esdmio_DMAWait* to wait for its completion or poll with *esdmio_DMAPoll* until *blocks_done* of 1 is returned. Only after one of these actions, a new DMA transfer may be started. The data will be found at the beginning of the DMA memory.

Driver version 1.1 and up:
If blocks parameter is a valid local FPGA address (doubly even, range 0..0x7FC), a transfer with incrementing local address is started instead of the usual transfer from fixed local 0x400.

This can be useful to get a register dump from the FPGA via DMA.

4.2.1.7 esdmio_DMAMultiStart

Syntax:

```
ESDMIO_RESULT esdmio_DMAMultiStart
(
 ESDMIO_HANDLE        hnd,
 ESDMIO_DMASETTINGS   *dmasetup
);
```

Description: Initiates a multi (continuous) DMA transfer of I/O data from the card to the DMA memory.

Arguments:

hnd must point to a valid ESDMIO_HANDLE.

dmasetup must point to a valid ESDMIO_DMASETTINGS.

For a multi transfer, the parameter blocks must be greater than 1.
The parameter blocksize must be a multiple of 16.
The transfer blocks will be at the beginning of the DMA memory, with no space between them. They form a cyclic buffer. Behind the data blocks, the descriptors are setup in DMA memory. The application must NOT touch these descriptors.
The size of the blocks (blocksize * blocks) plus the descriptors (blocks * descriptorsize) may not exceed the dma_size of the card.

The descriptorsize is 16 bytes for a 32 bit PCI address and 32 bytes for a 64 bit PCI access (means 32 bytes worst case).

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the DMA transfer has been started and is running in background.

Usage: It will never stop even if the handle is closed, it can only be stopped by calling esdmio_DMAMultiStop.

After starting a DMA transfer, one may either use *esdmio_DMAWait* to wait for the completion of at least one block or poll with *esdmio_DMAPoll*. The blocks_done then indicates the number of completed blocks.
The application must accumulate the blocks_done to keep track of the next buffer that will complete.

They will always complete in a consecutive way, wrapping to the first buffer after the last is filled.

4.2.1.8 esdmio_DMAMultiStop

Syntax:

```
ESDMIO_RESULT esdmio_DMAMultiStop
(
 ESDMIO_HANDLE  hnd
);
```

Description: Aborts a multi (continuous) DMA transfer.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the DMA transfer has been stopped and is not running in background any more.

It does not wait for a completed block, but stops the transfer also in the middle of a block. It waits for the stopping to be confirmed by the bridge.

4.2.1.9 esdmio_DMAPoll

Syntax:

```
ESDMIO_RESULT esdmio_DMAPoll
(
 ESDMIO_HANDLE        hnd,
 ESDMIO_DMASTATUS     *dmastatus
);
```

Description: Polls the status of a DMA transfer.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

dmastatus must point to an ESDMIO_DMASTATUS space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the **dmastatus* contains the number of completed blocks.

Each completed block will be reported only once.

4.2.1.10 esdmio_DMAWait

Syntax:

```
ESDMIO_RESULT esdmio_DMAWait
(
 ESDMIO_HANDLE        hnd,
 ESDMIO_DMASTATUS     *dmastatus
);
```

Description: Waits for the completion of (at least) one block of a DMA transfer.

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

dmastatus must point to an ESDMIO_DMASTATUS space.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the **dmastatus* contains the number of completed blocks. Each completed block will be reported only once.

4.2.1.11 IRQWait

Syntax:

```
ESDMIO_RESULT esdmio_IRQWait
(
 ESDMIO_HANDLE  hnd
);
```

Description: Waits for the assertion of an interrupt by the FPGA (non-DMA interrupt).

Arguments: *hnd* must point to a valid ESDMIO_HANDLE.

Return Values: If it returns with `ESDMIO_RESULT_SUCCESS`, the FPGA has asserted an interrupt. The interrupt is disabled until the next call of *esdmio_IRQWait*. The cause of the interrupt must be dealt with by the application.

If the interrupt cause remains active (e.g. not acknowledged), the next *esdmio_IRQWait* call will return immediately.

4.2.1.12 Return Codes

All functions return a status startíng with the prefix 'ESDMIO_RESULT_' which should always be evaluated by the application.

Success

If a call is terminated without errors, ESDMIO_RESULT_SUCCESS is returned.

No error

Value	Return code	Description
0	ESDMIO_RESULT_SUCCESS	The call was terminated without errors. The content of all returned values referenced by pointers are valid and must be evaluated by the application.

Error

In case of an error the following error codes might occur:

Value	Error code	Description
1	ESDMIO_RESULT_DEVICE_NOT_FOUND	device not found
2	ESDMIO_RESULT_DEVICE_BUSY	device busy
3	ESDMIO_RESULT_FAULTY_ADDRESS	faulty address
4	ESDMIO_RESULT_INVALID_PARAMETER	invalid parameter
5	ESDMIO_RESULT_OUT_OF_MEMORY	
6	ESDMIO_RESULT_OPERATION_INTERRUPTED	operation interrupted
7	ESDMIO_RESULT_DMA_NOT_RUNNING	DMA not running
88	ESDMIO_RESULT_OTHER_ERROR	other error

4.2.1.13 Example Source

```
/*

Copyright:       (C) 2011 esd electronic system design GmbH
Author:          Manuel Koeppen <manuel.koeppen@esd.eu>
                 Please report bugs to <support@esd.eu>
Purpose:         Example source for use of esdmio library

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public
License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

*/

#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#include <sys/mman.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>
#include <endian.h>
#include <byteswap.h>
#include <endian.h>
#if __BYTE_ORDER == __LITTLE_ENDIAN
#define le32_to_cpu(x) (x)
#define cpu_to_le32(x) (x)
#else
#define le32_to_cpu(x) bswap_32(x)
#define cpu_to_le32(x) bswap_32(x)
#endif

#include "esdmio.h"

#define DEBUG_FLUSH() do {fflush(stdout);fsync(STDOUT_FILENO);}while(0)
#define DEBUG_SLEEP() do {fflush(stdout);fsync(STDOUT_FILENO);usleep(100000);}while(0)
         /* 100ms for the debug output */

struct timeval t1,t2;
void timer_start(void) {
        gettimeofday(&t1,NULL);
}
void timer_stop(void) {
        gettimeofday(&t2,NULL);
}
unsigned int timer_usecs(void) {
        unsigned int x;
        x=t2.tv_sec-t1.tv_sec;
        x*=1000000;
        x+=t2.tv_usec;
        x-=t1.tv_usec;
        return x;
}

void print_dump(uint32_t *r, uint32_t len) {
        unsigned int i;
        if (r==NULL) return;
        for (i=0;i<(len>>2);i++) {
                if ((i&7)==0) printf("0x%04x:",i<<2);
                printf(" %08x",le32_to_cpu(r[i]));
                if ((i&7)==7) printf("\n");
        }
        DEBUG_SLEEP();
}

int esdmio_check_seq(uint32_t *buf, uint32_t len, int32_t *starting_seq)
{
        uint32_t i,i0,seq,nn,skp,ok=1;
         /* printf("esdmio_check_seq buf=%p len=0x%x seq=%d\n",buf,len,*starting_seq);DEBUG_SLEEP();return 0; */
        if (*starting_seq == -1) {
                /* ignore the first 256 entries, which might contain very old data */
                seq=(le32_to_cpu(buf[256])>>20)&0x7ff;
                i0=256;
        } else {
```

```
                seq=*starting_seq;
                i0=0;
        }
        nn=0;skp=0;
        printf("  Checking sequence counter...... ");
        DEBUG_FLUSH();
        for (i=i0;i<(len>>2);i++) {
                uint32_t b;
                b=le32_to_cpu(buf[i]);
                if ((b & 0xffff0000)==0xffff0000) {ok=0;skp++;continue;} /* skip empty entry */
                if (((b>>20)&0x7ff)!=seq) {
                        nn++;
                        printf("Invalid seq counter (expect 0x%03x is 0x%03x)at addr 0x%08x (%u)!\n",seq,
                               ((b>>20)&0x7ff),i*4,nn);
                        seq=((b>>20)+1)&0x7ff;
                        ok=0;
                } else {
                        seq=(seq+1)&0x7ff;
                        if ((nn!=0)&&(nn!=4)) printf(">>>>nn=%u\n",nn);
                        nn=0;
                }
        }
        if (skp) printf("%u empty entries skipped\n",skp);
        if (ok) printf("Ok\n");
        *starting_seq=seq;
        DEBUG_FLUSH();
        return ok;
}


int main(int argc, char **argv) {
        unsigned int i,minor;
        ESDMIO_RESULT ret;
        uint32_t *regs,*dma;
        ESDMIO_DMASETTINGS dmasettings;
        ESDMIO_DEVINFO devinfo;
        for (minor=0;minor<256;minor++) {
                ESDMIO_HANDLE h;
                if (esdmio_DeviceOpen(minor,&h)!=ESDMIO_RESULT_SUCCESS) continue;
                printf("Found card at index %u\n",minor);
                DEBUG_SLEEP();
                ret=esdmio_DeviceInfo(h,&devinfo);
                DEBUG_SLEEP();
                if (ret!=ESDMIO_RESULT_SUCCESS) {
                        printf("esdmio_DeviceInfo failed!\n");
                } else {
                        printf("Card is %s (type %u)\n",devinfo.devname,(unsigned int)devinfo.devtype);
                }
                DEBUG_SLEEP();
                ret=esdmio_MapRegisters(h,&regs);
                usleep(100000); /* 100ms for the debug output */
                if (ret!=ESDMIO_RESULT_SUCCESS) {
                        regs=NULL;
                        printf("esdmio_MapRegisters failed!\n");
                } else {
                        print_dump(regs,512);
                        regs[0x0E]=cpu_to_le32(0x11111111); /* set trigger mode to TimerA for all ADCs */
                        regs[0x08]=cpu_to_le32(0x00001062); /* about 1khz */
                        regs[0x0F]=cpu_to_le32(0x00000000); /* clear FIFO */
                        regs[0x14]=cpu_to_le32(0x7F0000FF); /* enable ADC1-8 data every sample, disable all
                                                               others */
                        regs[0x0F]=cpu_to_le32(0x10000000); /* DMA/FIFO enable timestamp=seq-count */
                }
                dmasettings.blocksize=0x10000;
                dmasettings.blocks=1;
                ret=esdmio_DMASingle(h,&dmasettings);
                DEBUG_SLEEP();
                if (ret!=ESDMIO_RESULT_SUCCESS) {
                        printf("esdmio_DMASingle failed!\n");
                } else {
                        ESDMIO_DMASTATUS dmastatus;
                        printf("dma running\n");
#if 0
                        for(i=0;;i++) {
                                ret=esdmio_DMAPoll(h,&dmastatus);
                                if (ret!=ESDMIO_RESULT_SUCCESS) {
                                        printf("esdmio_DMAPoll failed!\n");
                                        break;
                                }
                                /* printf("dmastatus.blocks_done=%u\n",(unsigned int)dmastatus.blocks_done); */
                                if (dmastatus.blocks_done) {
                                        printf("done after %u polls\n",i);
                                        break;
                                }
                        }
#else
                        ret=esdmio_DMAWait(h,&dmastatus);
                        if (ret!=ESDMIO_RESULT_SUCCESS) {
                                printf("esdmio_DMAWait failed!\n");
                                break;
```

```
                        }
                        printf("dmastatus.blocks_done=%u\n",(unsigned int)dmastatus.blocks_done);

#endif
                }
                ret=esdmio_MapDMA(h,&dma);
                DEBUG_SLEEP();
                if (ret!=ESDMIO_RESULT_SUCCESS) {
                        dma=NULL;
                        printf("esdmio_MapDMA failed!\n");
                } else {
                        print_dump(dma,256);
                }
                dmasettings.blocksize=0x2000;
                dmasettings.blocks=16;
                ret=esdmio_DMAMultiStart(h,&dmasettings);
                DEBUG_SLEEP();
                if (ret!=ESDMIO_RESULT_SUCCESS) {
                        printf("esdmio_DMAMultiStart failed!\n");
                } else {
                        ESDMIO_DMASTATUS dmastatus;
                        int32_t seq=-1;
                        uint32_t blk,t;
                        i=0;
                        blk=0;
                        printf("dma running\n");
                        do {
                                ret=esdmio_DMAWait(h,&dmastatus);
                                if (ret!=ESDMIO_RESULT_SUCCESS) {
                                        printf("esdmio_DMAWait failed!\n");
                                        break;
                                }
                                i+=dmastatus.blocks_done;
                                printf("dmastatus.blocks_done=%u total=%u\n",
                                         (unsigned int)dmastatus.blocks_done,i);
                                for(t=0;t<dmastatus.blocks_done;t++) {
                                        uint32_t *blk_addr;
                                        blk_addr=dma+
                                        ((dmasettings.blocksize>>2)*blk);
                                        if (esdmio_check_seq(blk_addr,dmasettings.blocksize,&seq)==0) {
                                             print_dump(blk_addr,dmasettings.blocksize);
                                        }
                                        blk=(blk+1)%dmasettings.blocks;
                                }
                                DEBUG_FLUSH();
                                if (i==7) { printf("Special sleep...");DEBUG_FLUSH();sleep(2);
                                             printf("done.\n");DEBUG_FLUSH(); }
                        } while(i<20);
                        ret=esdmio_DMAMultiStop(h);
                        if (ret!=ESDMIO_RESULT_SUCCESS) {
                                printf("esdmio_DMAMultiStop failed!\n");
                        }
                }
/* irq test */
                timer_start();
                regs[0x15]=cpu_to_le32(50000); /* 100ms */
                regs[0x0F]=cpu_to_le32(0x02040000); /* dma off TS=2us irq2(compare-irq bit 25) on */
                for(i=0;i<20;i++) {
                        uint32_t s;
                        ret=esdmio_IRQWait(h);
                        if (ret!=ESDMIO_RESULT_SUCCESS) {
                                printf("esdmio_IRQWait failed!\n");
                                break;
                        }
                        s=le32_to_cpu(regs[1]);
                        if (s & (1<<19)) { /* Compare_Irq */
                                regs[0x15]=cpu_to_le32(50000); /* 100ms */
                        }
                }
                timer_stop();
                i=timer_usecs();
                printf("Compare-Irq-Loop took %u us (%s)\n",i, ((i>1950000)&&(i<2050000))?"ok":"error");
#if 0
                regs[0x0F]=0;
                printf("Press Ctrl+C now...");DEBUG_FLUSH();
                for (;;) {
                        ret=esdmio_IRQWait(h);
                        if (ret!=ESDMIO_RESULT_SUCCESS) {
                                printf("esdmio_IRQWait failed (0x%x)!\n",ret);
                                break;
                        }
                }
#endif
                esdmio_DeviceClose(&h);
        }
        return 0;
}
```

4.3 Data Access

The AMC-ADIO24 offers simple I/O access via PCIe. But attention should be paid to the duration of a read cycle via PCIe, which can last up to 2...3 µs.

Therefore DMA operation to access the input process data, supported by the PCI bridge PEX8311, is highly recommended to achieve higher data rates. The DMA is assisted by a local FIFO in the FPGA, designed as a circular buffer with a depth of 256 entries.
For more information on the DMA mode read chapter 4.9 from page 83 on.

4.3.1 Direct I/O Access

A direct access to the I/Os can be achieved by FPGA registers. The following table displays examples of I/O registers for direct access:

I/O Port	Access	Register Name	See Page	Notes
Digital Inputs	read	DInAAct, DInBAct	69	read latched values of digital inputs
	read	DDI	41	read transparent values of digital inputs
Digital Outputs	write	DOWrite32	65	write all digital outputs at once
	write	DOWriteA, DOWriteB	65	write digital outputs
	read	DOutAAct, DOutBAct	69	read latched values of digital outputs
	read	DDO	42	read transparent values of digital outputs
Analog Inputs	read	ADC0Act … ADC7Act	68	read latched values of analog inputs
Analog Outputs	write	DACWriteBA	66	write analog outputs
	read	DACAAct, DACBAct	70	read latched values of analog outputs

The current timing of setting and reading the I/Os depends on the SYNC timing which strongly depends on the settings of of the so called 'Timing Routing Pool' (see page 32 et seqq.).

4.3.2 Setting Output Data

The output data is combined in six groups with 16 bits each.

Group	Outputs
1	Dout07 ... Dout00 (Enable + Data)
2	Dout15 … Dout08 (Enable + Data)
3	Dout23 … Dout16 (Enable + Data)
4	Dout31 … Dout24 (internal use only!)
5	AOUT1
6	AOUT2

Each of these groups has FIFOs with a depth of 256 entries. If a new entry is generated by the bus side (host CPU), the physical output is set with the next valid trigger condition.

A selective bit access to the digital outputs without reading the current output state can be executed by validation of a write access within an output group (see register 'DOWriteA/B').

4.3.3 Periodic Pattern Generation

The output-FIFO memory can be setup to generate periodical signals at the outputs (DOut-Pattern, SIN-Wave, …).
Is the according bit in parameter *Output-Table Mode* in register 'DIVMode' (reg. no. 0xF, page 53) set to '1', the last 256 entries of the FIFO are periodically output controlled by the according trigger event.

4.4 Streaming I/O and Trigger Selection

4.4.1 Streaming I/O Block Diagram

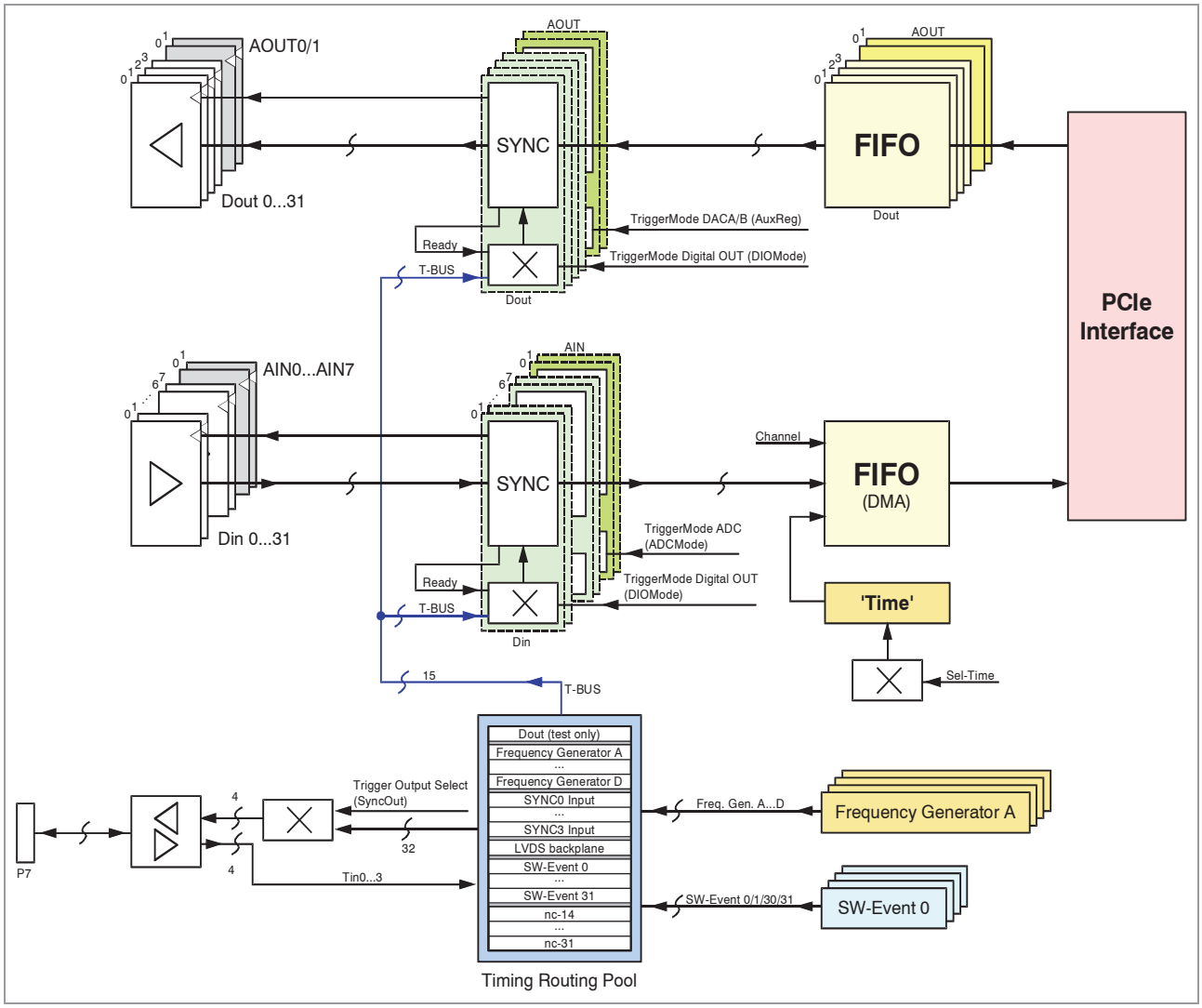


Figure 7: Streaming I/O block diagram

4.4.2 Trigger Selection

4.4.2.1 Overview

All input and output groups are featured with an independent trigger selector. For the outputs the trigger selector controls the reading of the FIFOs and the setting of the physical outputs. For the inputs the trigger selector controls the start of sampling.

For each group 16 different trigger sources are available:

0: channel-specific: 'done' generated next trigger ('free run')
1...15: trigger source is taken from Timing Routing Pool (1...15) 'rising edge'

4.4.2.2 Timing Routing Pool

The AMC-ADIO24 offers an internal routing pool for the trigger input and trigger output signals (T-Bus). This pool offers 32 independent signals. The trigger signals of the routing pool can be used to update the I/O signals and can be output at the SYNC0...SYNC3 interfaces as well.

Line	Trigger Source
0	DigOut(DORout) One of Dout31..Dout00 - For Test only
1	Frequency Generator A
2	Frequency Generator B
3	Frequency Generator C
4	Frequency Generator D
5	SYNC0 Input
6	SYNC1 Input
7	SYNC2 Input
8	SYNC3 Input
9	LVDSIn (Backplane, not yet implemented!)
10	SW-Event 0
11	SW-Event 1
12	SW-Event 30
13	SW-Event 31
14...31	reserved

Table 2: Assignment of the TRP (Timing Routing Pool)

To generate local trigger events the following options are available:

1. By software via generation of a software event
   (SW-Event 0, SW-Event 1, SW-Event 30, SW-Event 31; see page 59)

2. By timing generator (register FGENAB, FGENCD; see page 46)

3. By external trigger signal (SYNC0...SYNC3)

   Each of the 32 local timing signals can be output at the according 'SYNCX' channel.
   Where applicable the polarity can be inverted. All output signals are available as 'SYNCXin' in the routing pool as well.

4.5 Store/Restore Default Parameters (Calibration, FRU, FPGA-Image)

Beside the classic functions of an IPMI controller the controller at the AMC-ADIO24 offers some application-specific features:

1. Factory calibration data
   The analog inputs and outputs are calibrated at the factory during the end test of the module. During this calibration gain and offset values are determined and stored in the IPMI controller's EEPROM. At start-up the calibration data is loaded into the FPGA.
   The gain and offset values can be read and can be overwritten in the according FPGA registers (ADC0Corr...ADC7Corr, DACCor).
   Via a command register the user can store the current calibration data in the controller's EEPROM.

2. Reset to factory settings
   The calibration data, the FPGA-image and the FRU data can be reset to factory default via the command register.

A detailed register description is printed from page 80 on.

4.6 FPGA Memory Map

Addr. ... address
Attr. … attribute
Acc. … access
RO … read only (writes are ignored)
RW … read and write
R(W) … writable, but will be overwritten by local functions
WO … write only
b / w / l … permitted width for write access (b: byte, w: word, l: long)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x0	0x0000	RO	n/a	Version	abcdeeee	11000000	a = type, b = FPGA-version, cd = PCB-version, eeee = feature flags
0x1	0x0004	RO	n/a	Status	vvvvvvvv	xxxxxxxx	miscellaneous status bits
0x2	0x0008	RO	n/a	DDI	vvvvvvvv	xxxxxxxx	DirectDigitalIn 32 bit
0x3	0x000C	RO	n/a	DDO	vvvvvvvv	00000000	DirectDigitalOut 32 bit
0x4	0x0010	RW	bwl	DIOMode	hgfedcba	00000000	TriggerMode Digital In/Out
0x5	0x0014	RW	bwl	DOutInvert	vvvvvvvv	00000000	DigOut InversionMask 32 bit
0x6	0x0018	RW	bwl	HSE	vvvvvvvv	00000000	DigOut High_Side_Enable 32 bit
0x7	0x001C	RW	bwl	LSE	vvvvvvvv	00000000	DigOut Low_Side_Enable 32 bit
0x8	0x0020	RW	wl	FGENAB	bbbbaaaa	00000000	FreqGenA/B Value 2x16 bit
0x9	0x0024	RW	wl	FGENCD	ddddcccc	00000000	FreqGenC/D Value 2x16 bit
0xA	0x0028	RW	bwl	SyncOut	ddccbbaa	00000000	Trigger-Output-Select 4x8 bit (RS-485)
0xB	0x002C	RW	bwl	DORout	rrrraaee	00000000	aa = Selector DigOut to TRP0, ee = TriggerOut-Select (LVDS Backplane) rrrr = reserved
0xC	0x0030	RW	bwl	DInInvert	vvvvvvvv	00000000	DigIn InversionMask 32 bit
0xD	0x0034	RW	bwl	DInFilter	vvvvvvvv	00000000	DigIn FilterEnable 32 bit
0xE	0x0038	RW	bwl	ADCMode	hgfedcba	00000000	TriggerMode for ADC7...ADC0
0xF	0x003C	RW	bwl	DIVMode	missoxtt	00000000	m = DMA-Mode i = IrqEnable3...0 ss = TS-Counter-Select o = Master of DMA-Out x = unused tt = Output-Table Mode
0x10	0x0040	RW	bwl	AuxReg	rrrrdcba	00000000	ba = TriggerMode DACA/B, dc = Enable-Bits Out-FIFO-Irq 5...0 rrrr = reserved
0x11	0x0044	RO	n/a	QDac	bbbbaaaa	00000000	DirectDacValue DAC_B/A 2x16 bit
0x12	0x0048	RW	bwl	DACCorr	ddccbbaa	00000000	Gain/Offset-Correction DAC_B/A
0x13	0x004C	RW	bwl	EVENT	vvvvvvvv	00000000	PCI-Write: Set SW-Event 31...0 SPI-Write: Ack SPI-Irq All_Read: SPI-Irq 15...0, DMA-Pointer
0x14	0x0050	RW	bwl	DMAMode	ddddeeee	00000000	dddd = DMA-Write Disable eeee = DMA-TimeEnable 16-Bit

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description	
0x15	0x0054	R	-	Compare Value	ttttcccc	00000000	tttt = Current Timestamp Counter cccc = Current Compare Value	
		W	wl	Next	rrrrnnnn	00000000	nnnn = unsigned Delta Compare Value rrrr = reserved	
..	..	..	..	..	..	..	..	
0x30	0x00C0	RW	wl	ADC0Corr	uuuuggoo	ffff0000	ADC0: Gain/Offset-Correction gg = GainCorr, oo = OffsCorr uuuu = unused	
0x31	0x00C4	RW	wl	ADC1Corr	uuuuggoo	ffff0000	ADC1: see ADC0 above	
..	..	..	..	..	..	..	..	
0x37	0x00DC	RW	wl	ADC7Corr	uuuuggoo	ffff0000	ADC7: see ADC0 above	
0x38	0x00E0	RW	l	DOWrite32	vvvvvvvv	00000000	Write: 32 bit DigitalOut	Read: 32 bit DDO
0x39	0x00E4	RW	wl	DOWriteA	mmvvmmvv	00000000	Write: mask/value Dout 15...8 / Dout 7...0	Read: 32 bit DDO
0x3A	0x00E8	RW	wl	DOWriteB	mmvvmmvv	00000000	Write: mask/value Dout 31...24 / Dout 23...16	Read: 32 bit DDO
0x3B	0x00EC	RO	n/a	-----	---	---		Read: 32 bit DDO
0x3C	0x00F0	RW	wl	DACWriteBA	bbbbaaaa	00000000	Write: value DACB/DACA Read: 2x16 bit QDAC	
0x3D	0x00F4	..	..	reserved		rrrrrrrr		
0x3E	0x00F8	..	..	reserved		rrrrrrrr		
0x3F	0x00FC	RW	l	ChanWrite	rrrcvvvv	uuuuuuuu	write: value (channel) c = channel vvvv = value rrrr = reserved	
0x40	0x0100	R(W)	wl	ADC0Act	ttttvvvv	xxxxxxxx	ADC0 current value	
0x41	0x0104	R(W)	wl	ADC1Act	ttttvvvv	xxxxxxxx	ADC1 current value	
…	...	..	..	..	..	..	..	
0x47	0x011C	R(W)	wl	ADC7Act	ttttvvvv	xxxxxxxx	ADC7 current value	
0x48	0x0120	R(W)	wl	DInAAct	ttttvvvv	xxxxxxxx	Din 15...0 current value	
0x49	0x0124	R(W)	wl	DInBAct	ttttvvvv	xxxxxxxx	Din 31...16 current value	
0x4A	0x0128	R(W)	wl	DOutAAct	ttttvvvv	xxxxxxxx	Dout 15...0 current value	
0x4B	0x012C	R(W)	wl	DOutBAct	ttttvvvv	xxxxxxxx	Dout 31...16 current value	
0x4C	0x0130	R(W)	wl	DACAAct	ttttvvvv	xxxxxxxx	DACA current value	
0x4D	0x0134	R(W)	wl	DACBAct	ttttvvvv	xxxxxxxx	DACB current value	
0x4E	0x0138	R(W)	wl	TSAct	ttttvvvv	xxxxxxxx	Timestamp current value	
0x4F	0x013C	..	..	reserved		rrrrrrrr		
0x50	0x0140	R(W)	wl	ADC0Last	ttttvvvv	xxxxxxxx	ADC0 LastDMAValue	
0x51	0x0144	R(W)	wl	ADC1Last	ttttvvvv	xxxxxxxx	ADC1 LastDMAValue	
…	...	..	..	..	..	..	..	
0x57	0x015C	R(W)	wl	ADC7Last	ttttvvvv	xxxxxxxx	ADC7 LastDMAValue	
0x58	0x0160	R(W)	wl	DInALast	ttttvvvv	xxxxxxxx	Din 15...0 LastDMAValue	

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x59	0x0164	R(W)	wl	DInBLast	ttttvvvv	xxxxxxxx	Din 31...16 LastDMAValue
0x5A	0x0168	R(W)	wl	DOutALast	ttttvvvv	xxxxxxxx	Dout 15...0 LastDMAValue
0x5B	0x016C	R(W)	wl	DOutBLast	ttttvvvv	xxxxxxxx	Dout 31...16 LastDMAValue
0x5C	0x0170	R(W)	wl	DACALast	ttttvvvv	xxxxxxxx	DACA LastDMAValue
0x5D	0x0174	R(W)	wl	DACBLast	ttttvvvv	xxxxxxxx	DACB LastDMAValue
0x5E	0x0178	R(W)	wl	TSLast	ttttvvvv	xxxxxxxx	TimeStamp LastDMAValue
0x5F	0x017C	..	..	reserved		rrrrrrrr	
0x60	0x0180	RW	wl	ADC0Delta	ddddvvvv	00000000	ADC0 MinDelta for DMA
0x61	0x0184	RW	wl	ADC1Delta	ddddvvvv	00000000	ADC1 MinDelta for DMA
…	...	..	..	..	..	..	..
0x67	0x019C	RW	wl	ADC7Delta	ddddvvvv	00000000	ADC7 MinDelta for DMA
0x68	0x01A0	RW	wl	DInAMask	ddddvvvv	00000000	Din 15...0 Mask for DMA
0x69	0x01A4	RW	wl	DInBMask	ddddvvvv	00000000	Din 31...16 Mask for DMA
0x6A	0x01A8	RW	wl	DOutAMask	ddddvvvv	00000000	Dout 15...0 Mask for DMA
0x6B	0x01AC	RW	wl	DOutBMask	ddddvvvv	00000000	Dout 31...16 Mask for DMA
0x6C	0x01B0	RW	wl	DACADelta	ddddvvvv	00000000	DACA MinDelta for DMA
0x6D	0x01B4	RW	wl	DACBDelta	ddddvvvv	00000000	DACB MinDelta for DMA
0x6E	0x01B8	RW	wl	TSDelta	ddddvvvv	00000000	TS MinDelta for DMA
0x6F	0x01BC	RW	wl	unused	--------	00000000	unused
…	...	..	..	..	..	..	..
0x77	0x01DC	RW	wl	unused	--------	00000000	unused
0x78	0x01E0	RW	wl	Command	cccccccc	00000000	Command
0x79	0x01E4	RW	wl	Parameter	pppppppp	00000000	Parameter
0x7A	0x01E8	RW	wl	reserved		rrrrrrrr	
…	...	..	..	..	..	..	..
0x7E	0x01F8	RW	wl	reserved		rrrrrrrr	
0x7F	0x01FC	RO	n/a	Result	vvvvvvvv	00000000	Return code
0x80	0x0200	RW	wl	reserved		rrrrrrrr	
…	...	..	..	..	..	..	..
0xFF	0x03FC	RW	wl	reserved		rrrrrrrr	
0x100	0x0400	RO	n/a	DMARead0	tttcaaaa	xxxxxxxx	Read FIFO ttt = TimeStamp c = channel aaaa = value
0x101	0x0404	RO	n/a	DMARead1	tttcaaaa	xxxxxxxx	see above
	...	..	..	..	..	..	..
0x1FF	0x07FC	RO	n/a	DMARead...	tttcaaaa	xxxxxxxx	see above

4.7 Register Description

4.7.1 Version

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0	0x0000	RO	n/a	Version	abcdeeee	12000003	a= type, b= FPGA-version, cd= PCB-version, eeee= feature

Description

Variables	Name	Meaning
a	*FPGA_Type*	1: TCA_ADIO24 0, 2 … 15: reserved
b	*FPGA_Version*	1: Prototyp 2: Release 2 0, 3…15: reserved
cd	*PCB-Version*	PCB-Version (Is written by µC on startup.)
eeee	*Feature-Flags*	see table below

Feature-Flags Bits	Meaning
0	bit = 1: IRQ on FIFO-out
1	bit = 1: IRQ on Compare
2	reserved (bit = 0)
:	
15	reserved (bit = 0)

4.7.2 Status

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
1	0x0004	RO	n/a	Status	vvvvvvvv	xxxxxxxx	miscellaneous status bits

Description

Bit	Name		Meaning	
0	*FIFO_Empty*	*Dout7...0*	bit = 1: empty	
1	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
2			*FIFO_Status bit 1*	
3	*FIFO_Empty*	*Dout15...8*	bit = 1: empty	
4	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
5			*FIFO_Status bit 1*	
6	*FIFO_Empty*	*Dout23...16*	bit = 1: empty	
7	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
8			*FIFO_Status bit 1*	
9	*FIFO_Empty*	*Dout31...24*	bit = 1: empty	
10	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
11			*FIFO_Status bit 1*	
12	*FIFO_Empty*	*DACA*	bit = 1: empty	
13	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
14			*FIFO_Status bit 1*	
15	*FIFO_Empty*	*DACB*	bit = 1: empty	
16	*FIFO_Status*		*FIFO_Status bit 0*	*meaning of FIFO_Status* bits see table at page 40
17			*FIFO_Status bit 1*	
18	*Irq_Status0*		bit = 1: active	
19	*Irq_Status1*		bit = 1: active	
20	*Irq_Status2*		bit = 1: active	
21	*Irq_Status3*		bit = 1: active	
22...24	unused		'0'	
25	*DMAFIFFULL*		bit = 1: FIFO to local_bus Full	
26	*DMAFIFEMPTY*		bit = 1: FIFO to local_bus Empty	
27	*RS485InA*		Status of Input Signal (Test_only)	
28	*RS485InB*		Status of Input Signal (Test_only)	
29	*RS485InC*		Status of Input Signal (Test_only)	
30	*RS485InD*		Status of Input Signal (Test_only)	
31	*LVDSIn*		Status of Input Signal (Test_only)	

The following table shows the meaning of the *FIFO_Status* bits:

FIFO_Status		FIFO #Data Bytes	State
bit 1	*bit 0*		
0	0	0 ≤ #Data < 64	almost empty
0	1	64 ≤ #Data < 224	in range
1	0	224 ≤ #Data < 255	almost full
1	1	#Data = 255	FIFO full

Table 3: *FIFO_Status* bits

4.7.3 DDI (Direct Digital In, Inclusive Inversion)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
2	0x0008	RO	n/a	DDI	vvvvvvvv	xxxxxxxx	DirectDigitalIn 32-bit

Description
Read back of unsampled digital input port.

Bit no.	Digital Input
0	Din0
1	Din1
...	...
31	Din31

Din31 … Din24 are internally hard wired to Dout31 .... Dout24.

4.7.4 DDO (Direct Digital Out, No Inversion)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
3	0x000C	RO	n/a	DDO	vvvvvvvv	00000000	DirectDigitalOut 32-bit

Description
Current output state.

Bit no.	Digital Output
0	Dout0
1	Dout1
...	...
31	Dout31

Dout31 … Dout24 are internally hard wired to Din31 .... Din24.

4.7.5 DIOMode (Trigger Mode Din/Dout)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
4	0x0010	RW	bwl	DIOMode	hgfedcba	00000000	TriggerMode Digital In/Out

Description
Selection of trigger signals for digital input/output groups.

Bit Assignment

Bit no.	Content of Register DIOMode	Mode
3 … 0	a	*Mode_Dout 7...0*
7 … 4	b	*Mode_Dout 15...8*
11 … 8	c	*Mode_Dout 23...16*
15 … 12	d	*Mode_Dout 31..24*
19 … 16	e	*Mode_Din 7...0*
23 … 19	f	*Mode_Din 15...8*
27 … 24	g	*Mode_Din 23...16*
31 … 28	h	*Mode_Din 31...24*

Meaning of Parameter *Mode_...*:

Value of *Mode_...*	Meaning
0	'own trigger': trigger next when ready
15 … 1	TRP 15...1 (timing routing pool line 15...1)

4.7.6 DOut Invert (Digital Output Inverted)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
5	0x0014	RW	bwl	DOutInvert	vvvvvvvv	00000000	DigOut InversionMask 32-bit

Description

Bit no.	Digital Output
0	bit = 1: *invert physical output Dout0*
1	bit = 1: *invert physical output Dout1*
...	...
31	bit = 1: *invert physical output Dout31*

4.7.7 HSE/LSE (HighSide Enable / LowSide Enable)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
6	0x0018	RW	bwl	HSE	vvvvvvvv	00000000	DigOut High_Side_Enable 32-bit
7	0x001C	RW	bwl	LSE	vvvvvvvv	00000000	DigOut Low_Side_Enable 32-bit

Description
Configuration of digital out characteristic.

Bit no. of HSE or LSE	Assigned to Digital Output
0	Dout0
1	Dout1
...	...
31	Dout31

HSE and LSE are evaluated together for each physical output:

Register Bits			Pin	Details	
HSEx	LSEx	Doutx			
0	0	x	open	I/O is input only	
1	0	0	open	High Side Switch:	OFF
1	0	1	5V	High Side Switch:	ON
0	1	0	open	Low Side Switch:	OFF
0	1	1	GND	Low Side Switch:	ON
1	1	0	GND	Push/Pull:	OFF
1	1	1	5V	Push/Pull:	ON

4.7.8 FGENAB (DDFS Frequency Generator)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
8	0x0020	RW	wl	FGENAB	bbbbaaaa	00000000	FreqGenA/B Value 2x16 bit

Description

The DDFS frequency generator generates a square wave signal with a 50:50 duty cycle.

aaaa: parameter values of frequency generator A
bbbb: parameter values of frequency generator B

$$\text{Frequency} = \frac{32\ \text{MHz}}{\textit{Scaler}} \cdot \textit{DDFS\_Value}$$

Bit no. of aaaa and bbbb	Parameter
15, 14	*Scaler*
13 … 0	*DDFS_Value*

Bit no.		Value of *Scaler*
15	14	
0	0	$2^{(15+12)}$
0	1	$2^{(15+9)}$
1	0	$2^{(15+6)}$
1	1	$2^{(15+3)}$

Range of Frequency
0 ... 3.906 kHz
0 … 31.25 kHz
0 … 250 kHz
0 … 2 MHz

4.7.9 FGENCD (Clock Divider)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
9	0x0024	RW	wl	FGENCD	dddcccc	00000000	FreqGenC/D Value 2x16 bit

Description

The frequency generator generates a square wave signal with a 50:50 duty cycle.

`cccc`: *Scaler* of frequency generator C
`dddd`: *Scaler* of frequency generator D

```
                16 MHz
Frequency = ------------
                Scaler
```

Note:
Do not use *Scaler* < 16!

4.7.10 SyncOut (External Trigger Output/Input)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xA	0x0028	RW	bwl	SyncOut	ddccbbaa	00000000	Trigger-Output-Select 4x 8 bits (RS-485)

Description
Selects the source for SyncOut0...3 .

Bytes of Register SyncOut	Trigger Output Select
aa	*TrigOut_SYNC0*
bb	*TrigOut_SYNC1*
cc	*TrigOut_SYNC2*
dd	*TrigOut_SYNC3*

Bits of *TrigOut_SYNCx*	Meaning
4 … 0	source selector 0...31, selects TRP (Timing Routing Pool signal; see table at page 33)
5	bit = 1: Invert Output
6	bit = 1: Invert Input
7	bit = 1: Enable Output

4.7.11 DORout (Trigger Output/Input, DigOut-Route)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xB	0x002C	RW	bwl	DORout	rrrraaee	00000000	aa= Selector DigOut to TRP0, ee= TriggerOut-Select (LVDS Backplane) rrrr=reserved

Description
Multiplexes a discrete line to TRP0.

Bytes of Register DORout	Meaning
aa	*Selector_DigOut_to_TRP0*
ee	*TriggerOut-Select LVDS*
rrrr	*reserved*

Bits of Parameter *Selector_DigOut_to_TRP0*	Meaning
4 … 0	Selector Digital Out 31...0 (Dout31...0) to TRP0
5	reserved, write 0
6	reserved, write 0
7	reserved, write 0

Parameter *TriggerOut-Select LVDS* (**DO NOT USE!**):
Bit 7...0: SyncOut for LVDS (backplane) transceiver.

Bits of *TriggerOut-Select LVDS*	Meaning
4 … 0	source selector 31...0, selects TRP (Timing Routing Pool signal; see table at page 33)
5	bit = 1: Invert Output
6	bit = 1: Invert Input
7	bit = 1: Enable Output

Attention: The routing via backplane is untested!

4.7.12 DInInvert (Digital Input Inverted)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xC	0x0030	RW	bwl	DInInvert	vvvvvvvv	00000000	DigIn InversionMask 32 bit

Description

Bit no.	Digital Input
0	bit = 1: invert physical input Din0
1	bit = 1: invert physical input Din1
...	...
31	bit = 1: invert physical input Din31

4.7.13 DInFilter (Filter Digital Input)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xD	0x0034	RW	bwl	DInFilter	vvvvvvvv	00000000	DigIn FilterEnable 32 bit

Description

Bit no.	Digital Input
0	bit = 1: enable filter on physical input Din0
1	bit = 1: enable filter on physical input Din1
...	...
31	bit = 1: enable filter on physical input Din31

Value of *enable filter on physical input DinX* = 1 : filter enabled
(3 successive samples set/reset state)

4.7.14 ADCMode (Trigger Mode ADC7...0)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xE	0x0038	RW	bwl	ADCMode	hgfedcba	00000000	Trigger Mode for ADC7 … ADC0

Description
Selects trigger condition for ADC7...0.

Nibbles of register ADCMode	Meaning
a	*TriggerMode ADC0*
b	*TriggerMode ADC1*
c	*TriggerMode ADC2*
d	*TriggerMode ADC3*
e	*TriggerMode ADC4*
f	*TriggerMode ADC5*
g	*TriggerMode ADC6*
h	*TriggerMode ADC7*

Format of *TriggerMode ADCx* see Register 'DIOMode' at page 43.

4.7.15 DIVMode (DMA-Mode, Timer Select, Interrupt Enable)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0xF	0x003C	RW	bwl	DIVMode	missoxtt	00000000	m = DMA-Mode i = IrqEnable3...0 ss = TS-Counter-Select o = Master of DMA-Out x = unused tt = Output-Table Mode

Description

Content of Register DIVMode	Width	Meaning
m	4 bits	*DMA-Mode*
i	4 bits	*IrqEnable3...0*
ss	8 bits	*TS-Counter Select*
o	4 bits	*Master of DMA-Out*
x	4 bits	unused
tt	8 bits	*Output-Table Mode*

DMA-Mode
Bits 31...28 of register DIVMode are used for the selection of the *DMA-Mode.*

Value of parameter *DMA-Mode*	(Value of bits 31...28 of DIVMode)	Meaning
0	0000	DMA-disabled (clear read- and write-counter)
1	0001	DMA-enabled (FIFO-mode)
2	0010	Memory-Mode, one-shot
3	0011	Memory-Mode, continuous
4...15	0010...1111	reserved

IrqEnable3...0
Bits 27...24 of register DIVMode are used for *IrqEnable3...0.*

Bits of parameter *IrqEnable 0...3*	(Bits of DIVMode)	Meaning	Usage
0	24	bit = 1: Enable Irq0	FIFO-Irq
1	25	bit = 1: Enable Irq1	Compare-Irq
2	26	bit = 1: Enable Irq2	unused
3	27	bit = 1: Enable Irq3	unused

TS-Counter Select *(Timestamp Counter Select)*
Bits 18...16 of register DIVMode are used for *TS-Counter Select*.
Bits 23...19 of register DIVMode are reserved for future use.

Value of parameter *TS-Counter Select*	(Value of bits 18...16 of DIVMode)	Timestamp counter resolution
0	000	any write to DMA-FIFO (sequence counter)
1	001	0.25 µs
2	010	0.5 µs
3	011	1 µs
4	100	2 µs
5	101	4 µs
6	110	8 µs
7	111	128 µs

Master of DMA-Out
Bits 14...12 of register DIVMode are used for *Master of DMA-Out*
Bit 15 of register DIVMode is reserved for future use.

Value of parameter *Master of DMA-Out*	(Value of bits 14...12 of DIVMode)	Selected DMA master
0	000	no DMA-Out
1	001	Dout 08...01 is master
2	010	Dout 16...09 is master
3	011	Dout 24...17 is master
4	100	Dout32...25 is master
5	101	DACA is master
6	110	DACB is master
7	111	DACB is master (reserved)

Output-Table Mode
Bits 5...0 of register DIVMode are used for *Output-Table Mode*
Bits 11...6 of register DIVMode are reserved for future use.

Bits of parameter *Output-Table Mode*	(Bits of DIVMode)	Assignment of the Outputs
0	0	Dout 08...01
1	1	Dout 16...09
2	2	Dout 24...17
3	3	Dout32...25
4	4	DACA
5	5	DACB

*V*alue of *Output-Table Mode* Bits	Meaning
0	use Output-FIFO as FIFO
1	output always the last 256 entries ('Table-Mode')

4.7.16 Auxreg (Trigger Mode DAC A/B)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x10	0x0040	RW	bwl	AuxReg	rrrrdcba	00000000	ba= Trigger Mode DACA/B dc= Enable bits Out-FIFO-Irq 5...0

Description

Bytes of Register AuxReg	Meaning
ba	*Trigger Mode DACA/B*
dc	*En_Out-FIFO-Irq*
rrrr	reserved

Bits of Parameter *TriggerMode DACA/B*	Meaning
3 … 0	*Trigger Mode_DACA*
7 … 4	*Trigger Mode_DACB*

Format of *Mode_DACA/B* see register 'DIOMode' on page 43.

Bits of Parameter *En_Out-FIFO-Irq*	(Bits of AuxReg)	Meaning
0	8	bit = 1: Enable Irq on FIFO Dout 7...0 *)1
1	9	bit = 1: Enable Irq on FIFO Dout 15...8 *)1
2	10	bit = 1: Enable Irq on FIFO Dout 23...16 *)1
3	11	bit = 1: Enable Irq on FIFO Dout 31...24 *)1
4	12	bit = 1: Enable Irq on FIFO DACA *)1
5	13	bit = 1: Enable Irq on FIFO DACB *)1

*)1 active in FIFO status = '00' (empty/almost empty) see register 'Status' at page 39.
These interrupts are routed to IRQ0.

4.7.17 QDac (Direct Read Back DAC A/B)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x11	0x0044	RO	n/a	QDac	bbbbaaaa	00000000	DirectDacValue Corrected DAC_B/A 2x16 bit

Description

QDac returns the physical DAC-value after gain/offset correction (Out_Value).

Words of Register QDac	Meaning
aaaa	*DirectDacValue DACA (16 bit)*
bbbb	*DirectDacValue DACB (16 bit)*

$$\text{Vout} = 10.24\text{V} \cdot \frac{\textit{Value}}{0x8000}$$

For the DAC-value range the converter components require the data in 'BTC' (Binary Tow's Complement) format. The following table shows the assignment of the data to output voltages (Vout) for this format (signed 16-bit) by means of some basic data.

Value	**Vout**	
0x8000	- 10,24 V	(- maximum value)
:	:	
0xFFFF	- 0.3125 mV	(- 1 LSB)
0x0000	0	
0x0001	+ 0.3125 mV	(+1 LSB)
:	:	
0x7FFF	+10.24 V - (1 LSB) = +10,2397 V	(+ maximum value)

4.7.18 DACCorr (Gain/Offset Correction DAC B/A)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x12	0x0048	RW	bwl	DACCorr	ddccbbaa	00000000	Gain/Offset-Correction DAC_B/A

INFORMATION
The analog outputs are factory calibrated. The calibration data is loaded to this FPGA register at every start-up.

Description

The user can change the factory calibration data at the register DACCorr.
The changed contents of DACCorr are lost at reset or power-off. For permanent storage the contents of DACCorr can be stored in a EEPROM. Restoration of the factory calibration is possible as well. For details please refer chapter '4.7.29 Command Register' from page 80 on.

Bytes of Register DACCorr	Meaning
aa	*Offset DAC_A*
bb	*Gain DAC_A*
cc	*Offset DAC_B*
dd	*Gain DAC_B*

Correction formula:

$$\text{Out\_Value} = (\text{In\_Value} + \textit{Offset}) \cdot \left(1 + \frac{\textit{Gain}}{2^{14}}\right)$$

Correction coefficients are signed (signed 8 bit [-128, ... 0, ... 127]).

Out_Value is limited to 0x7FFF / 0x8000 on overflow.

4.7.19 EVENT

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x13	0x004C	RW	bwl	EVENT	rrrrvvvv	00000000	PCI-Write: Set SW-Event 31...0 SPI-Write: Ack SPI-Irq All_Read: SPI-Irq status 15...0

Description
The function of the register EVENT depends on the write or read access type.

PCI-Write

Bit no.	Meaning
0	bit = 1: generate SW-Event 0
1	bit = 1: generate SW-Event 1
...	...
16	bit = 1: generate SW-Event 16
...	...
31	bit = 1: generate SW-Event 31

SPI-Write (internal use only)

Bit no.	Meaning
0	bit = 1: Ack SPI-Irq0
1	bit = 1: Ack SPI-Irq1
...	...
15	bit = 1: Ack SPI-Irq15
16	bit = 1: SW-Event16
...	...
31	bit = 1: SW-Event31

Read

Bit no.	Meaning
15 … 0	SPI-Irq 15...0
31 … 16	reserved

Event Assignment

Event	Source
SW-Event 0	Source for TRP(10)
SW-Event 1	Source for TRP(11)
SW-Event 2...7	reserved
SW-Event 8	SPI-Irq8 (Command-Irq, see page 80)
...	...
SW-Event 15	SPI-Irq15 (future use)
SW-Event 16...29	reserved
SW-Event 30	Source for TRP(12)
SW-Event 31	Source for TRP(13)

SW-Event 0, 1, 30, 31 are generating a 250 ns pulse on the trigger lines.

4.7.20 DMAMode (DMA enable/disable)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x14	0x0050	RW	bwl	DMAMode	dddeeee	00000000	dddd = DMA-Write Disable eeee = DMA-TimeEnable 16-Bit

Description

Content of Register DMAMode	Width	Meaning
dddd	16 bits	*DMA-Write Disable*
eeee	16 bits	*DMA-Time Enable 16-Bit*

Combined evaluation of *DMA-Write Disable* (dddd) and *DMA-Time Enable 16-Bit* (eeee);

bits 'd' (of *DMA-Write Disable)*	bits 'e' (of *DMA-Time Enable 16-Bit)*	Write to DMA FIFO	Update xxxAct	Cycles
0	0	on valid Event	always	7
0	1	always	always	4
1	0	never	never	0
1	1	never	always	4

For more details about the combined evaluation of *DMA-Write Disable* (dddd) and *DMA-Time Enable 16-Bit* (eeee) see page 82.

Each of the 16 bits of the parameters is assigned to one DMA channel (see following table).

DMA-Write Disable (dddd), ***DMA-Time Enable 16-Bit*** (eeee)

The 16 bits of the parameters are assigned to the DMA channels:

Bits of parameter *DMA-Write Disable* 'dddd'	Bits of parameter *DMA-Time Enable 16-Bit* 'eeee'	Assigned to DMA channel
0	0	ADC0
1	1	ADC1
2	2	ADC2
3	3	ADC3
4	4	ADC4
5	5	ADC5
6	6	ADC6
7	7	ADC7
8	8	Din 15...0
9	9	Din 31...16
10	10	Dout 15...0
11	11	Dout 31...16
12	12	DACA
13	13	DACB
14	14	TimeStamp
15	15	unused

Table 4: Assignment of bits of *DMA-Write Disable* and *DMA-Time Enable 16-Bit* to DMA channel

4.7.21 Compare

Timestamp-Compare-Irq
The lower 16 bits of the Timestamp Counter (*Current Timestamp Counter)* are compared with a Compare Value (*Current Compare Value*). If they match, the local Irq #1 is set.

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x15	0x0054	R	-	Compare Value	`ttttcccc`	`00000000`	tttt = Current TS Counter cccc = Current Compare Value
		W	bwl	Next	`rrrrnnnn`	`00000000`	nnnn = unsigned Delta Compare Value

Description

Content of Register Compare	Width	Meaning
`tttt`	16 bits	*Current Timestamp Counter*
`cccc`	16 bits	*Current Compare Value*
`rrrr`	16 bits	*reserved*
`nnnn`	16 bits	*Delta Compare Value*

On Read: ***Current Timestamp Counter***
Bits 31...16 of register 'Compare' return the *Current Timestamp Counter*. These bits contain the current lower 16 bits of the Timestamp Counter.

Current Compare Value
Bits 15...0 of register 'Compare' return the *Current Compare Value*.

On Write: Acknowledge the pending compare interrupt.

Current_Compare_Value = Current_Compare_Value + Delta_Compare_Value

4.7.22 ADC0Corr ... ADC7Corr (ADC Gain and Offset)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x30	0x00C0	RW	wl	ADC0Corr	uuuuggoo	0000	ADC0: Gain/Offset-Correction gg=GainCorr, oo=OffsCorr
0x31	0x00C4	RW	wl	ADC1Corr	uuuuggoo	0000	ADC1: see above
..	..	..	..	..	..	..	..
0x37	0x00DC	RW	wl	ADC7Corr	uuuuggoo	0000	ADC7: see above

INFORMATION
The analog inputs are factory calibrated. The calibration data is loaded to these FPGA registers at every start-up.

Description
The user can change the factory calibration data at the registers ADC0Corr...ADC7Corr.
The changed contents of the ADCxCorr registers are lost at reset or power-off. For permanent storage the contents of the ADCxCorr registers can be stored in a EEPROM. Restoration of the factory calibration is possible as well. For details please refer chapter '4.7.29 Command Register' from page 80 on.

Bytes of Register ADCxCorr	Meaning
oo	*Offset*
gg	*Gain*
uuuu	unused

Correction formula:

$$\text{Out\_Value} = (\text{In\_Value} + \textit{Offset}) \cdot \left(1 + \frac{\textit{Gain}}{2^{14}}\right)$$

Correction coefficients are signed.

Out_Value is limited to 0x7FFF / 0x8000 on overflow.

4.7.23 DOWriteX (Digital Out Mask)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description	
0x38	0x00E0	RW	l	DOWrite32	vvvvvvvv	00000000	Write: 32 bit DigitalOut	Read: 32 bit DDO
0x39	0x00E4	RW	wl	DOWriteA	mmvvmmvv	00000000	Write: Mask/Value Dout15...08/Dout07...00	Read: 32 bit DDO
0x3A	0x00E8	RW	wl	DOWriteB	mmvvmmvv	00000000	Write: Mask/Value Dout31...24/Dout23...16	Read: 32 bit DDO

Description

The setting of the digital outputs of a channel group is always done via the according FIFOs:

Group	Outputs
1	Dout07...Dout00
2	Dout15...Dout08
3	Dout23...Dout16
4	Dout31...Dout24

Within the group the eight according mask/value data are stored. Triggered by the according timing signal (DIOMode) the physical outputs are set.

DOWrite32 sets all 32 outputs at once.

DOWriteA/B expects mask and value in two 16-bit data words.
For setting an output bit, the related mask bit must be '1'.
If a mask bit = '0', the output bit stays unchanged.

4.7.24 DACWriteBA (Write DAC)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x3C	0x00F0	RW	wl	DACWriteBA	bbbbaaaa	00000000	Write: Value DACB/DACA Read: 2x16 bit DACA/BAct

Description

Words of Register DACWriteBA	Meaning
aaaa	*Value DACA*
bbbb	*Value DACB*

The values for DACA/DACB have to be written in the according FIFO.

Triggered by the according timing signal (AuxReg) the data correction is executed and the physical output of the DAC is set.

$$Vout = 10.24V \cdot \frac{Value}{0x8000}$$

Please refer to page 57 for further information about the data format of Vout.

4.7.25 ChanWrite

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x3F	0x00FC	WO	I	ChanWrite	rrrcvvvv	uuuuuuuu (undefined)	write: Value(Channel)

Description
An alternative method for writing output data (DMA-write).

Content of Register ChanWrite	Meaning
c	*Channel*
vvvv	*Value*
rrr	reserved

Value of *Channel*	Meaning
0x0	Group0 (Dout07...Dout00)
0x1	Group1 (Dout15...Dout08)
0x2	Group2 (Dout23...Dout16)
0x3	Group3 (Dout31...Dout24)
0x4	DACA
0x5	DACB
0x6...0xF	do not use

Value of *Value*	Meaning
0x0000...0xFFFF	*Output Value Group 0 (Dout07...Dout00)*
0x0000...0xFFFF	*Output Value Group 1 (Dout15...Dout08)*
0x0000...0xFFFF	*Output Value Group 2 (Dout23...Dout16)*
0x0000...0xFFFF	*Output Value Group 3 (Dout31...Dout24)*
0x0000...0xFFFF	*Output Value DACA*
0x0000...0xFFFF	*Output Value DACB*

For further description of the digital output values Dout00 ... Dout31 see DOWriteX on page 65. As for DOWriteX the 16-bit consist of mask and value.
For description of DACA/DACB see page 66.

4.7.26 Current Register Values

The current register value can be read directly at any time, but attention should be paid to the duration of a read cycle via PCIe, which can last up to 2...3 µs !
Therefore the operation via DMA is highly recommended to achieve higher data rates.

Write access is possible. After the following trigger event the new values will be written.

Note:
The data of the registers described in this chapter will be written in the DMA-FIFO, if DMA mode is configured.

4.7.26.1 ADCxAct (Current Analog Input Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x40	0x0100	R(W)	wl	ADC0Act	tttttvvvv	xxxxxxxx	ADC0 current value
0x41	0x0104	R(W)	wl	ADC1Act	tttttvvvv	xxxxxxxx	ADC1 current value
…	...	..	..	..	..	..	..
0x47	0x011C	R(W)	wl	ADC7Act	tttttvvvv	xxxxxxxx	ADC7 current value

Description

Content of Register ADCxAct	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (current ADC value)

$$Vin = 10.24V \cdot \frac{Value}{0x8000}$$

For the ADC-value range the converter components require the data in 'BTC' (Binary Tow's Complement) format. The following table shows the assignment of the data to output voltages (Vin) for this format (signed 16-bit) by means of some basic data.

Input	**Vin**	
0x8000	- 10,24 V	(- maximum value)
:	:	
0xFFFF	- 0.3125 mV	(- 1 LSB)
0x0000	0	
0x0001	+ 0.3125 mV	(+1 LSB)
:	:	
0x7FFF	+10.24 V - (1 LSB) = +10,2397 V	(+ maximum value)

4.7.26.2 DInAAct, DInBAct (Current Digital Input Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x48	0x0120	R(W)	wl	DInAAct	ttttvvvv	xxxxxxxx	Din15...0 current value
0x49	0x0124	R(W)	wl	DInBAct	ttttvvvv	xxxxxxxx	Din31...16 current value

Description

Content of Register DInA/B Act	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (current digital input value)

4.7.26.3 DOutAAct, DOutBAct (Current Digital Output Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x4A	0x0128	R(W)	wl	DOutAAct	ttttvvvv	xxxxxxxx	Dout15..0 current value
0x4B	0x012C	R(W)	wl	DOutBAct	ttttvvvv	xxxxxxxx	Dout31..16 current value

Description

Content of Register DOutA/B Act	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (current digital output value)

4.7.26.4 DACAAct, DACBAct (Current Analog Output Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x4C	0x0130	R(W)	wl	DACAAct	tttvvvv	xxxxxxxx	DACA current value
0x4D	0x0134	R(W)	wl	DACBAct	tttvvvv	xxxxxxxx	DACB current value

Description

Content of Register DACA/B Act	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (current analog output value)

Please refer to page 57 for further information about the data format of Vout.

4.7.26.5 TSAct (Current Timestamp Counter Value)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x4E	0x0138	R(W)	wl	TSAct	tttvvvv	xxxxxxxx	TimeStamp current value

Description

Content of Register TSAct	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (upper 16 bits of the 27-bit TS-counter)

4.7.27 Last DMA Values

4.7.27.1 ADC0Last... ADC7Last (Last Analog Input DMA Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x50	0x0140	R(W)	wl	ADC0Last	tttvvvv	xxxxxxxx	ADC0 LastDMAValue
0x51	0x0144	R(W)	wl	ADC1Last	tttvvvv	xxxxxxxx	ADC1 LastDMAValue
…	...	..	..	..	. .	. .	..
0x57	0x015C	R(W)	wl	ADC7Last	tttvvvv	xxxxxxxx	ADC7 LastDMAValue

Description

Last value written to DMA-In FIFO.

Content of Register ADCxLast	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (ADC last DMA value)

Please refer to page 68 for further information about the data format of Vin.

4.7.27.2 DInALast, DInBLast (Last Digital Input DMA Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x58	0x0160	R(W)	wl	DInALast	ttttvvvv	xxxxxxxx	Din15...0 LastDMAValue
0x59	0x0164	R(W)	wl	DInBLast	ttttvvvv	xxxxxxxx	Din31...16 LastDMAValue

Description

Last value written to DMA-In FIFO.

Content of Register DInA/B Last	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (digital input last DMA value)

4.7.27.3 DOutALast, DOutBLast (Last Digital Output DMA Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x5A	0x0168	R(W)	wl	DOutALast	ttttvvvv	xxxxxxxx	Dout15...0 LastDMAValue
0x5B	0x016C	R(W)	wl	DOutBLast	ttttvvvv	xxxxxxxx	Dout31...16 LastDMAValue

Description

Last value written to DMA-In FIFO.

Content of Register DOutA/B Last	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (digital output last DMA value)

4.7.27.4 DACALast, DACALast (Last Analog Output DMA Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x5C	0x0170	R(W)	wl	DACALast	ttttvvvv	xxxxxxxx	DACA LastDMAValue
0x5D	0x0174	R(W)	wl	DACBLast	ttttvvvv	xxxxxxxx	DACB LastDMAValue

Description

Last value written to DMA-In FIFO.

Content of Register DACA/B Last	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (analog output last DMA value)

Please refer to page 57 for further information about the data format of Vout.

4.7.27.5 TSLast (Last TS-Counter DMA Values)

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x5E	0x0178	R(W)	wl	TSLast	ttttvvvv	xxxxxxxx	TimeStamp Last DMA Value

Description

Last value written to DMA-In FIFO.

Content of Register TSLast	Meaning
tttt	*TS_counter* (lower 16 bits of the 27-bit TS-counter)
vvvv	*Value* (upper 16 bits of the 27-bit TS-counter)

4.7.28 Difference Values for Trigger Conditions

4.7.28.1 ADC0Delta … ADC7Delta

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x60	0x0180	RW	wl	ADC0Delta	ddddvvvv	00000000	ADC0 MinDelta for DMA
0x61	0x0184	RW	wl	ADC1Delta	ddddvvvv	00000000	ADC1 MinDelta for DMA
…	...	..	..	..	. .	. .	..
0x67	0x019C	RW	wl	ADC7Delta	ddddvvvv	00000000	ADC7 MinDelta for DMA

Description

Content of Register ADCxDelta	Meaning
dddd	do not care (always write '0')
vvvv	*Value* (minimum difference value of analog input)

In the registers xxxxDelta the trigger conditions for writing the data in the DMA-FIFO are defined:

Trigger = ABS(ADCxAct - ADCxLast) > ADCxDelta

For more information see DMA-Mode from page 86 on.

Please refer to page 68 for further information about the data format of Vin.

4.7.28.2 DInA/BMask

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x68	0x01A0	RW	wl	DInAMask	dddvvvv	00000000	Din15..0 Mask for DMA
0x69	0x01A4	RW	wl	DInBMask	dddvvvv	00000000	Din31..16 Mask for DMA

Description

Content of Register DInA/B Mask	Meaning
dddd	do not care (always write '0')
vvvv	*Value* (mask values for digital inputs)

In the registers DInA/BMask the trigger conditions for writing the data in the DMA-FIFO are defined:

Trigger = (DInxAct EXOR DInxLast) AND NOT(DInxMask) ≠ 0x0000

For more information see DMA-Mode from page 86 on.

4.7.28.3 DOutA/BMask

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x6A	0x01A8	RW	wl	DOutAMask	dddvvvv	00000000	Dout15...0 Mask for DMA
0x6B	0x01AC	RW	wl	DOutBMask	dddvvvv	00000000	Dout31...16 Mask for DMA

Description

Content of Register DOutA/B Mask	Meaning
dddd	do not care (always write '0')
vvvv	*Value* (mask values for digital outputs)

In the registers DOutA/BMask the trigger conditions for writing the data in the DMA-FIFO are defined:

Trigger = (DOutxAct EXOR DOutxLast) AND NOT(DOutxMask) ≠ 0x0000

For more information see DMA-Mode from page 86 on.

4.7.28.4 DACA/BDelta

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x6C	0x01B0	RW	wl	DACADelta	dddvvvv	00000000	DACA MinDelta for DMA
0x6D	0x01B4	RW	wl	DACBDelta	dddvvvv	00000000	DACB MinDelta for DMA

Description

Content of Register DACA/B Delta	Meaning
dddd	do not care (always write '0')
vvvv	*Value* (minimum difference value of analog output)

In the registers DACA/BDelta the trigger conditions for writing the data in the DMA-FIFO are defined:

Trigger = ABS(DACxAct - DACxLast) > DACxDelta

For more information see DMA-Mode from page 86 on.

Please refer to page 57 for further information about the data format of Vout.

4.7.28.5 TSDelta

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x6E	0x01B8	RW	wl	TSDelta	dddvvvv	00000000	TS MinDelta for DMA

Description

Content of Register TSDelta	Meaning
dddd	do not care (always write '0')
vvvv	*Value* (minimum difference value of analog output)

In the register TSDelta the trigger condition for writing the data in the DMA-FIFO are defined:

Trigger = ABS(TSAct - TSLast) > TSDelta

For more information see DMA-Mode from page 86 on.

Please refer to page 57 for further information about the data format of Vout.

4.7.29 Command Register

Reg No	Addr.	Attr.	Write Acc.	Name	Format	Default	Description
0x78	0x01E0	RW	wl	Command	cccccccc	00000000	Command
0x79	0x01E4	RW	wl	Parameter	pppppppp	00000000	Parameter
0x7A	0x01E8	RW	wl	reserved		rrrrrrrr	
…	...	..	..	..	..	..	..
0x7E	0x01F8	RW	wl	reserved		rrrrrrrr	
0x7F	0x01FC	RO	n/a	Result	vvvvvvvv	00000000	Return code

Description

Setting **Command** Register	Defines (examples)	Parameter required?	Command
0	AMC_ADIO24CMD_NOP	no	no operation, returns OK
1	AMC_ADIO24CMD_FRU_RESTORE	no	restore factory FRU information
2	AMC_ADIO24CMD_FRU_CLEAR	no	delete FRU information
3	AMC_ADIO24CMD_CAL_READ	no	read calibration data of analog inputs and outputs from EEPROM and write to FPGA register
4	AMC_ADIO24CMD_CAL_SAVE	no	read calibration data of analog inputs and outputs from FPGA registers and write to EEPROM
5	AMC_ADIO24CMD_CAL_RESTORE	no	restore factory defaults of calibration data and write them to EEPROM and FPGA register
6	AMC_ADIO24CMD_ALL_RESTORE	no	restore factory default state of FPGA-image, FRU and calibration
0x07...0x40	unused	-	-
0x41...0x4F	reserved	reserved	reserved for factory service purposes
0x50... 0xFFFFFFFF	unused	-	-

Return Codes of **Result** Register	Defines (examples)	Status message
0	AMC_ADIO24CMD_RESULT_OK	OK
0xFFFFFFFF	AMC_ADIO24CMD_RESULT_INVALIDCMD	invalid command
0xFFFFFFFE	AMC_ADIO24CMD_RESULT_INVALIDPARA	invalid parameter
0xFFFFFFFD	AMC_ADIO24CMD_RESULT_NOCALIBDATA	no valid calibration data found

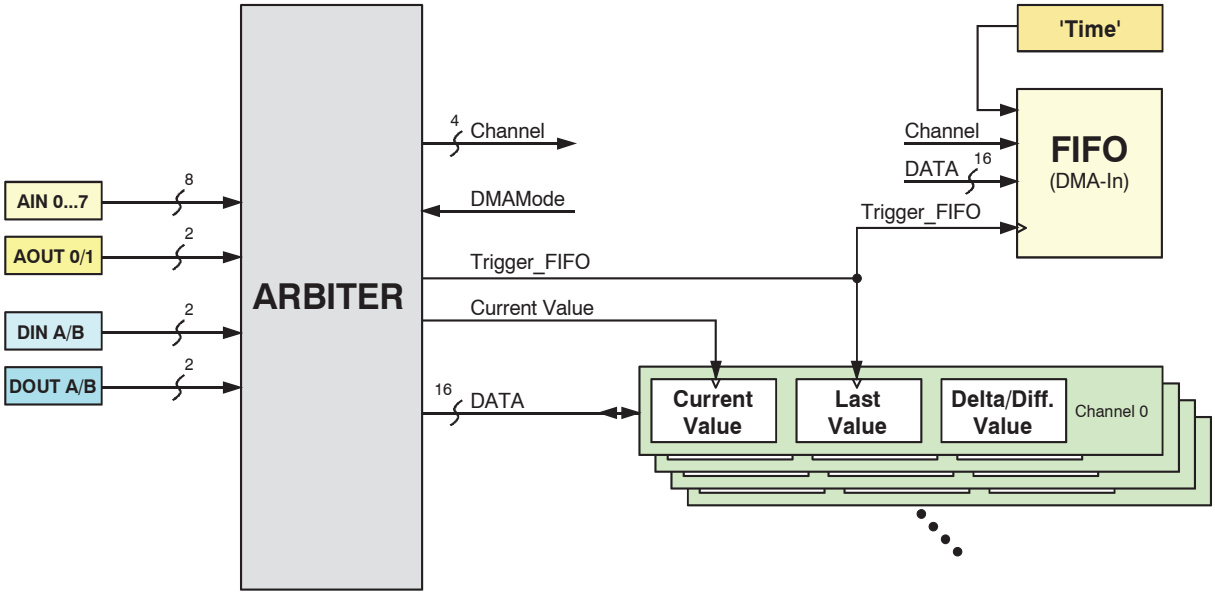
Command Setting Procedure:

The execution of a command is controlled by the firmware. In order to check the success of the command's execution the register EVENT has to be polled:

1. Write command value to register 'Command' (register no. 0x78)
2. Set bit 8 in register 'EVENT' (register no. 0x13) (e.g. by writing value 0x100)
3. Poll register 'EVENT' until value of bit 8 is returned to '0' (i.e. firmware is ready)
4. Evaluate result code from register 'Result' (register no. 0x7F)

4.8 I/O Data Processing

Each I/O group generates a Ready signal after a trigger event occurs. These Ready signals are routed to an arbiter, which carries out the data processing according to signal priority (e.g. support of 'xxxACT', resp. check of valid DMA-In conditions).



The following maximum occurring band width is required:

ADC: 8 x 200 KSPS = 1.2 MSPS
DAC: 2 x 600 KSPS = 1.2 MSPS
DIN: 2 x 1 MSPS = 2.0 MSPS
DOUT: 2 x 1 MSPS = 2.0 MSPS

This results in an average data rate of 6.4 MSPS, resp. 25.6 MB/s.

A complete processing of all input requests (check of DMA-In conditions) without loss can be guaranteed up to an average data rate of approx. 4 MSPS.
Therefore the following data processing can be configured (via bits 'e' and 'd' in register 'DMAMode'; channel assignment as described in chapter '4.7.20 DMAMode (DMA enable/disable)' from page 61 on).

bits 'd' (of *DMA-Write Disable)*	bits 'e' (of *DMA-Time Enable 16-Bit)*	Write to DMA FIFO	Update xxxAct	Cycles
0	0	on valid Event	always	7
0	1	always	always	4
1	0	never	never	0
1	1	never	always	4

To guarantee a loss-free processing, the following condition must be valid:

Sum of all channels (Sample_rate(ch) • Cycles(ch,mode)) << 32 MHz

4.9 DMA Access

4.9.1 DMA Input Data

4.9.1.1 Read Data FIFO

If a valid DMA-FIFO trigger condition occurs ('Trigger_FIFO'), the data is stored in combination with time stamp and channel number in a 32-bit wide, 256 entries deep, FIFO. From the FIFO the DMA unit of the PCI bridge can transfer the entries to the CPU memory.

If at least 16 entries are stored in the FIFO (FPGA to PCI) the signal 'DREQ0' at the PCI bridge becomes active and remains active as long as more than 4 entries are stored in the FIFO.

Memory Mode: Read of address returns data of write pointer. No incrementation of read pointer (for test purpose only).

FIFO Mode: A read of any FIFO address returns FIFO data and increments the read pointer if the FIFO is not empty.

Read Data FIFO Memory Area

Addr.	Attr.	Acc.	Name	Format	Default	Description	
0x0400	RO	n/a	DMARead0	`tttcaaaa`	`xxxxxxxx`	Read FIFO	ttt= TimeStamp c= Channel aaaa= Value
0x0404	RO	n/a	DMARead1	`tttcaaaa`	`xxxxxxxx`	see above	
...	..	..	..	..	..	..	
0x07FC	RO	n/a	DMARead...	`tttcaaaa`	`xxxxxxxx`	see above	

Description

Content of Register DMAReadx	Meaning
`ttt`	*Timestamp* (11+1 bits)
`c`	*DMA channel no* (4 bits)
`aaaa`	*Value* (16 bits)

Timestamp Generation

To enable timestamps > 12 bits the following logic is implemented:

The local timestamp counter (TS counter) is realized as a combination of a 11-bit counter (lower count) and a 16-bit counter (upper count).
Within the 12-bit area of the process data timestamp the 11-bit count value plus a 'pending' flag is transmitted. This 'pending' flag is set, if the local 11-bit counter overruns and is reset, if the 16-bit counter is written in the FIFO.
If trigger conditions apply simultaneously, the 16-bit counter (DMA-channel = 0xE) gets the worst priority.
To determine timestamp values > 11 bit, the application just has to execute the following operation:

```
TimeStampLong = LocalUpperCount<<11 + DMAData>>20
```

If the DMA-channel = 0xE, the LocalUpperCount is taken.

DMA Channel No

DMA channel no	I/O Group
0x0	ADC0
0x1	ADC1
0x2	ADC2
0x3	ADC3
0x4	ADC4
0x5	ADC5
0x6	ADC6
0x7	ADC7
0x8	Din15..0
0x9	Din31..16
0xA	Dout15..0
0xB	Dout31..16
0xC	DACA
0xD	DACB
0xE	TimeStamp
0xF	FIFO-empty

4.9.1.2 Setting DMA Operation

The DMA operation mode is configured via the register DIVMode (see page 53):

0. DMA disabled

read-adress = write-adress = 0, DMA disabled

1. DMA-enabled (FIFO-mode)

A valid trigger condition causes an entry in the circular buffer and the write address (WA) is incremented as long as WA-RA < 255 (RA = read address).

Each long word read access in the range of 0x0400 … 0x07FC returns the content of RA. RA is incremented as long as WA-RA>0. If the FIFO is empty (WA=RA) the read access returns 0xFFFFwwrr (ww=WA, rr=RA).

2. Memory-Mode, one-shot

Writing data as in FIFO-mode (1.), but the read address is determined from the access address. The write address can not exceed 255 (One-Shot-Mode).

3. Memory-Mode, continuous

As 2., but the write address wraps from 255 to 0.

For more information on the DMA at PCIe read the manual of the PCI bridge PEX 8311 (http://www.plxtech.com/products/expresslane/pex8311).

4.9.2 DMA Output Data

Instead of CPU writes output data can be set by the DMA controller of the PCI bridge (e.g if digital-out patterns or analog wave forms are required).

For this the second DMA channel ('DREQ1') of the PCI bridge can be used. The FIFO state of one of the output groups can be evaluated as the DMA requester (see parameter *Master of DMA-Out* in register DIVMode (reg. no. 0xF)).

DREQ1 is set according to the following conditions:

DREQ1 is active,	if the FIFO state is '00' (< 64 entries in FIFO)
DREQ1 stays active,	as long as the FIFO state is '01' (< 244 entries in the FIFO)

The according register 'DOWriteA/B', resp. 'DACWriteA/B' can be used as write addresses.
If several output groups are controlled BY THE SAME TRIGGER LINE, they can be supported by the same DMA. Thererfore one of the output channels has to be select as 'master'. Note, that the data of the 'slave' channels are included in the DMA stream, as well.
The register 'Chanwrite (reg. no. 0x3F) has to be used as write address.

For more information on the DMA at PCIe read the manual of the PCI bridge PEX 8311 (http://www.plxtech.com/products/expresslane/pex8311).

5. Technical Data

5.1 General Technical Data

Power supply voltage	nominal voltage: 3.3 V ($I_{3.3VMPMAX}$ = 60 mA), 12 V ($I_{12VTYPICAL}$ = 0.6 A, I_{12VMAX} = 1.0 A)
Connectors	AMC-ADIO24 only: AOUT0, AOUT1 (10-pin har-link® connector, P1) - 2x analog output AIN0 ... AIN3 (10-pin har-link® connector, P2) - 4x analog input AIN4 ... AIN7 (10-pin har-link® connector, P3) - 4x analog input DIO0 ... DIO7 (10-pin har-link® connector, P4) - 8x digital I/O DIO8 ... DIO15 (10-pin har-link® connector, P5) - 8x digital I/O DIO16 ... DIO23 (10-pin har-link® connector, P6) - 8x digital I/O SYNC0 ... SYNC3 (10-pin har-link® connector, P7) - 4x RS-485, trigger/sync AMC-ADIO24-HD50 only: AOUT0, AOUT1 (50-pin har.mik® connector, P1) - 2x analog output, AIN0 ... AIN7 8x analog input, DIO0 ... DIO23 24x digital I/O SYNC0 ... SYNC3 (10-pin har-link® connector, P2) - 4x RS-485, trigger/sync AMC-ADIO24/-HD50: AMC plug connector (170-pin AMC plug connector, J1) - AMC B/B+ compatible (MicroTCA™) Only for test- and programming purposes: X600 programming, debugging X1850 JTAG interface
Temperature range	0...70 °C ambient temperature (free convection)
Humidity	max. 90%, non-condensing
Dimensions	Single Mid-size AdvancedMC Module (73.8x18.96x181.5 mm)
Weight	140 g

Table 5: General data of the module

5.2 MicroTCA™ /AMC Standards

µTCA	PICMG® MTCA.0 R1.0, PICMG® AMC.0 R2.0
IPMI	IPMI V1.5
Updates	PICMG® HPM.1 R1.0
PCIe	PCISIG® PCIe spec. R.1.0a, only lane 4 is used
Connector	AMC plug connector according to PICMG® 3.0 Rev. 3.0 AdvancedTCA® Base Specification and PICMG® AMC.1 R2.0 PCI Express on AdvancedMC™

Table 6: MicroTCA standards

5.3 Analog Outputs

Number of outputs	2 outputs, available as single-ended and differential (AOUT0, AOUT1)
Output voltage	U_{OUT} = ±10 V
Resolution	16 bit
Sampling rate	up to 600 KSPS
Output current	I_{OUTMAX} = 10 mA
Output state if board is switched off	high-resistance
I/O control	FPGA Spartan II
Protection circuits	transient protection U_{BREAK} ≈ ±26 V
Connector	AMC-ADIO24: 10-pin har-link female connector (P1) AMC-ADIO24-HD50: (50-pin har.mik® connector, P1)

Table 7: Data of analog outputs

5.4 Analog Inputs

Number of inputs	8 inputs, (AIN0 ... AIN7)
Input voltage	U_{IN} = ±10 V absolute max. input voltage with respect to Analog_GND: U_{INMAX} = ±13 V
Resolution	16 bit
Sampling rate	up to 200 KSPS
Input filter	Low pass filter (Bessel filter 3rd order), f_G = 50 kHz
Input resistance	R_{IN} = 100 kΩ
I/O control	FPGA Spartan II
Protection circuits	transient protection U_{BREAK} ≈ ±26 V
Connector	AMC-ADIO24: 10-pin har-link female connector (P2, P3) AMC-ADIO24-HD50: (50-pin har.mik® connector, P1)

Table 8: Data of analog inputs

5.5 Digital Inputs/Outputs

Digital I/Os	24 digital I/Os (DIO0 ... DIO23), TTL-level, each channel separately programmable as • input only • output sink • output source • output sink/source
Maximum output current/channel	64 mA sink, 32 mA source
Input circuit	comparator with hysteresis, threshold '1' → '0' : U_{OFF}= 0.8 V threshold '0' → '1' : U_{ON} = 2.4 V
Input resistance	Status = '0' : R_{in0} ≈ 4 kΩ to GND Status = '1' : R_{in1} ≈ 4 kΩ to +5V
I/O control	FPGA Spartan II
Input sampling rate	up to 1 MHz
Output update rate	up to 1 MHz
I/O protection	short circuit protection $U_{OUT} = f(I_{OUT})$ transient protection circuit activated at negative voltages below: $U_{LOW} < -0.7V$ activated at positive voltages from: $U_{HIGH} > +7.0V$
Connector	AMC-ADIO24: 10-pin har-link female connectors (P4, P5, P6) AMC-ADIO24-HD50: (50-pin har.mik® connector, P1)

Table 9: Data of digital I/Os

5.6 Trigger Ports

Number	4x RS-485, trigger/sync (SYNC0 ... SYNC3)
Data rate	max. 1 MHz
Connector	AMC-ADIO24: 10-pin har-link female connector (P7) AMC-ADIO24-HD50: (50-pin har.mik® connector, P2)

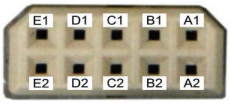
Table 10: Data of the trigger ports

6. Connector Assignment of AMC-ADIO24

6.1 Analog Out

Device connector: 10-pin har-link female connector, P1,
For the position of the connector in the front panel see page 14.

Pin Position:



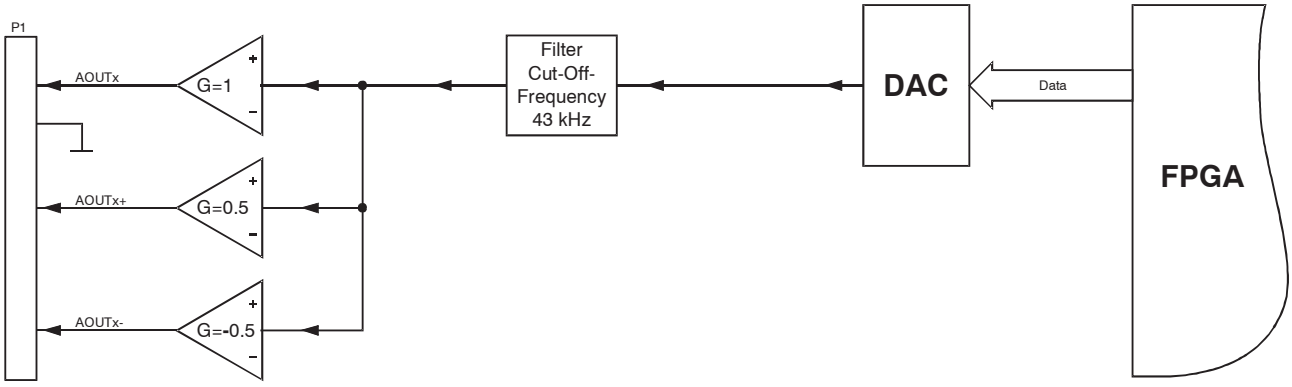
Pin Assignment:

AOUT0, AOUT1 (P1)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	AOUT1+	AOUT0+	-	AOUT1	AOUT0
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	AOUT1-	AOUT0-	ANALOG_GND	AOUTGND1	AOUTGND0

Signal Description:

AOUTx+, AOUTx-	analog output signal lines of differential output circuit of channel x (x = 0, 1)
AOUTx, AOUTxGND	analog output signal lines of single ended output circuit of channel x (x = 0, 1)
ANALOG_GND	reference potential of analog I/Os

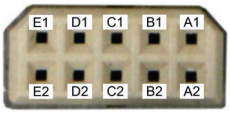
Block Diagram Output Circuit:



6.2 Analog In

Device connector: 10-pin har-link female connector, P2, P3
For the position of the connectors in the front panel see page 14.

Pin Position:



Pin Assignment:

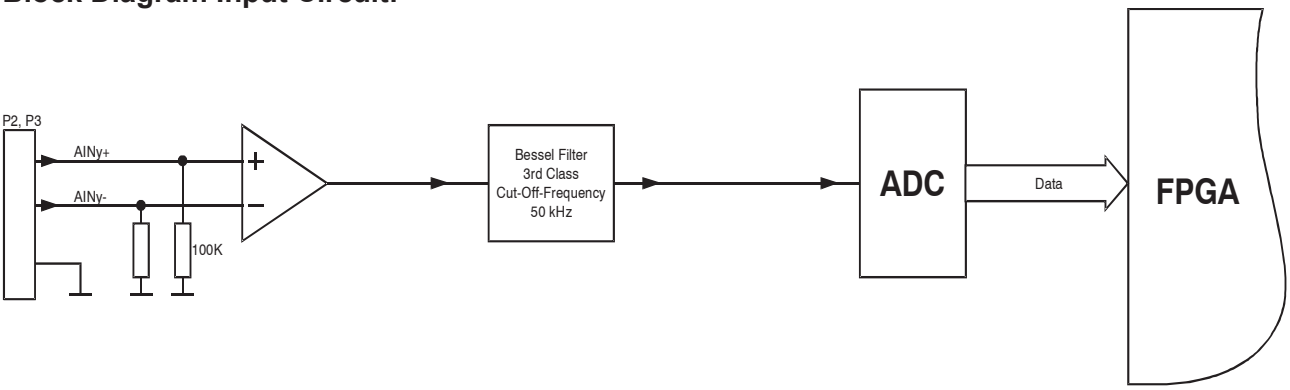
AIN0 ... AIN3 (P2)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	AIN3+	AIN2+	-	AIN1+	AIN0+
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	AIN3-	AIN2-	ANALOG_GND	AIN1-	AIN0-

AIN4 ... AIN7 (P3)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	AIN7+	AIN6+	-	AIN5+	AIN4+
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	AIN7-	AIN6-	ANALOG_GND	AIN5-	AIN4-

Signal Description:

AIN+, AINx-	analog input signal lines of channel x (x = 0 ...7)
ANALOG_GND	reference potential of analog I/Os

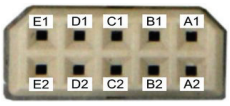
Block Diagram Input Circuit:



6.3 Digital I/O

Device connector: 10-pin har-link female connector, P4, P5, P6
For the position of the connectors in the front panel see page 14.

Pin Position:



Pin Assignment:

DIO0 ... DIO7 (P4)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	DIO6	DIO4	-	DIO2	DIO0
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	DIO7	DIO5	GND	DIO3	DIO1

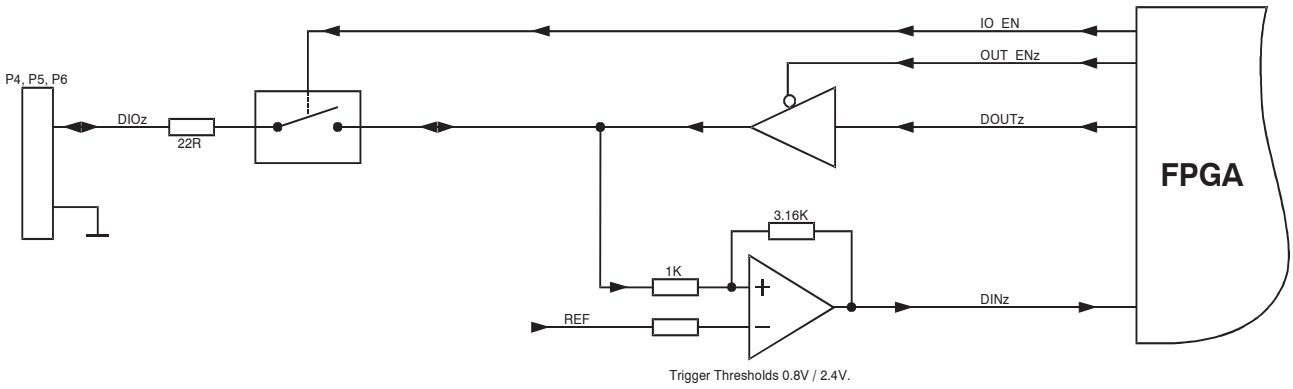
DIO8 ... DIO15 (P5)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	DIO14	DIO12	-	DIO10	DIO8
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	DIO15	DIO13	GND	DIO11	DIO9

DIO16 ... DIO23 (P6)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	DIO22	DIO20	-	DIO18	DIO16
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	DIO23	DIO21	GND	DIO19	DIO17

Signal Description:

DIOx digital input/output signal lines of channel x (x = 0 ...23)
GND reference potential of digital I/Os

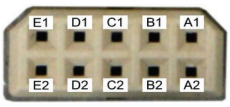
Block Diagram I/O-Circuit:



6.4 Trigger Ports

Device connector: 10-pin har-link female connector, P7
For the position of the connector in the front panel see page 14.

Pin Position:



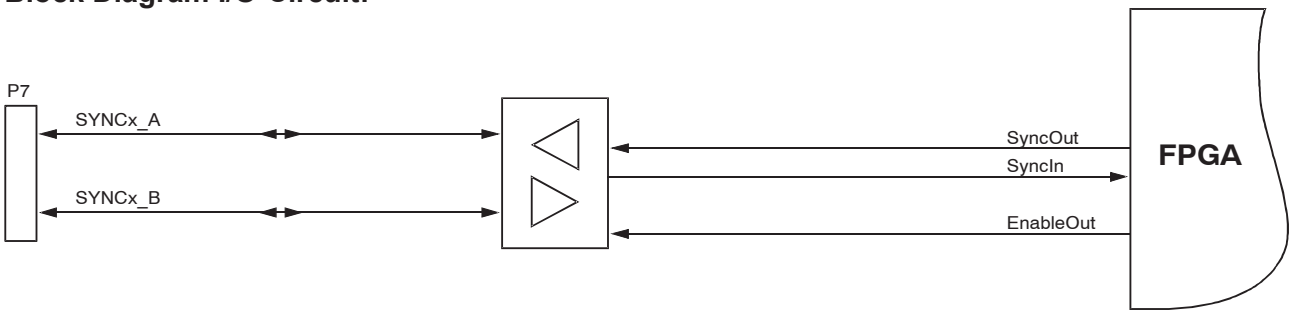
Pin Assignment:

SYNC0 ... SYNC3 (P7)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	SYNC3_A	SYNC2_A	-	SYNC1_A	SYNC0_A
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	SYNC3_B	SYNC2_B	GND	SYNC1_B	SYNC0_B

Signal Description:

SYNCx_A,
SYNCx_B signal lines of the RS-485 ports (trigger/sync) of channel x (x = 0 ... 3)
GND reference potential of trigger/sync I/Os

Block Diagram I/O-Circuit:



7. AMC-ADIO-Adapter for AMC-ADIO24

INFORMATION
The AMC-ADIO24-Adapter (order No.: U.1001.10) is intended for the usage with the AMC-ADIO24 (U.1001.01).

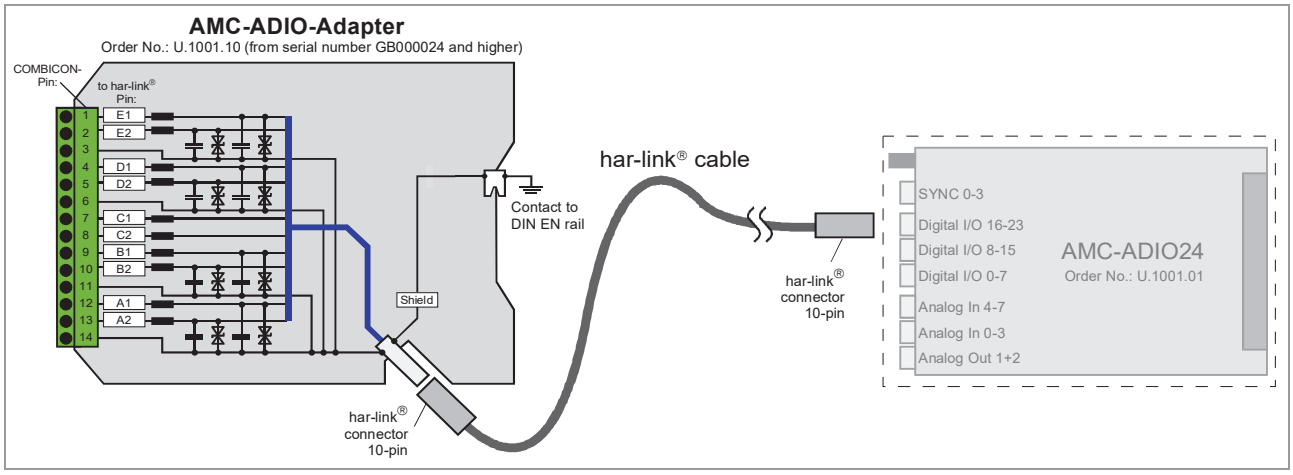


Figure 8: AMC-ADIO- Adapter

The AMC-ADIO-Adapter is designed to support the wiring of the AMC-ADIO24. The adapter can be connected to each I/O-type of the AMC-ADIO24: digital I/O, analog inputs, analog outputs or SYNC-ports.

Eight signal lines plus Ground and Shield are routed via the har-link® cable to the Mini COMBICON connector at the AMC-ADIO-Adapter. One Shield clamp is available for a signal pair. This ensures optimum shielding of the cable assembly.
The shield potential is automatically connected with the DIN-EN rail via a spring cage contact when the AMC-ADIO-Adapter is mounted on the rail.

If connected to one of the analog input connectors at the AMC-ADIO24, four differential analog I/O signals plus Shield and Ground are available. Connected to the analog output connector two analog outputs each as differential and single ended plus Shield are available.
If connected to one of the digital I/O connectors eight digital I/O-signals plus Shield and Ground are available. Connected to the SYNC-connector 4 SYNC ports plus Shield are available.

HF ferrite in the signal lines improve the EMC properties of the AMC-ADIO-Adapter.

The Mini COMBICON connector is pluggable to allow easy removing of the cable assemblies.

7.1 AMC-ADIO-Adapter View

Figure 9: AMC-ADIO-Adapter view with COMBICON connector

7.2 Technical Data AMC-ADIO-Adapter

Connectors	AMC side: (10-pin har-link® connector, female) - routed signals Process side: (14-pin COMBICON style connector with spring-cage connection)
Temperature range	-40 °C ... +85 °C ambient temperature
Humidity	max. 90%, non-condensing
Dimensions	22.5 mm x 112 mm x 113 mm
IP Class	IP20
UL-Flame Rating	UL94 V-0
Weight	tbd.
Connectors	AMC side: 10-pin har-link® female connector Process side: 14-pin COMBICON style connector FK-MCP 1,5/14-STF-3,81

Table 11: General data of the adapter

7.3 Connector Assignment

Eight signal lines plus ground and shield are routed via the har-link® cable to the Mini COMBICON connector at the AMC-ADIO-Adapter.

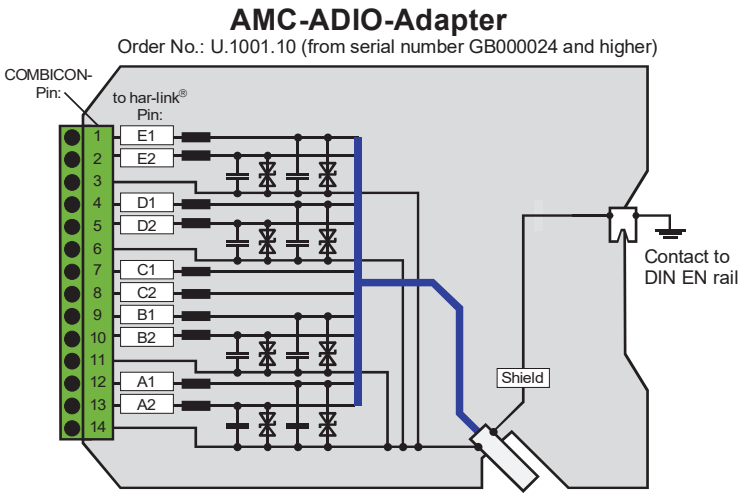


Figure 10: Signal assignment of the 14-pin COMBICON connector

Pin	Signal routed	Signal
1	E1	XSIG0+
2	E2	XSIG0-
3	Shield_E	Shield
4	D1	XSIG1+
5	D2	XSIG1-
6	Shield_D	Shield
7	C1	Reserved
8	C2	XAGND0
9	B1	XSIG2+
10	B2	XSIG2-
11	Shield_B	Shield
12	A1	XSIG3+
13	A2	XSIG3-
14	Shield_A	Shield

Signal Description

XSIGy+, XSIGy- Signal lines of the connected interface routed via the har-link® cable to the Mini-COMBICON connector at the AMC-ADIO-Adapter, channel y (y= 0, 1, 2, 3) The assignment of the pins depends on the assignment of the AMC-ADIO24 har-link connector connected (see page 87).

Pin	Signal routed	Pin assignment depending on the AMC-ADIO24 connector connected to the AMC-ADIO-Adapter						
		P1 Analog Out	**P2** Analog In	**P3** Analog In	**P4** Digital IO	**P5** Digital IO	**P6** Digital IO	**P7** Trigger Ports
1	E1	AOUT1+	AIN3+	AIN7+	DIO6	DIO14	DIO22	SYNC3_A
2	E2	AOUT1-	AIN3-	AIN7-	DIO7	DIO15	DIO23	SYNC3_B
3	Shield	Shield	Shield	Shield	Shield	Shield	Shield	Shield
4	D1	AOUT0+	AIN2+	AIN6+	DIO4	DIO12	DIO20	SYNC2_A
5	D2	AOUT0-	AIN2-	AIN6-	DIO5	DIO13	DIO21	SYNC2_B
6	Shield	Shield	Shield	Shield	Shield	Shield	Shield	Shield
7	C1	-	-	-	-	-	-	-
8	C2	ANALOG _GND	ANALOG_ GND	ANALOG_ GND	GND	GND	GND	GND
9	B1	AOUT1	AIN1+	AIN5+	DIO2	DIO10	DIO18	SYNC1_A
10	B2	AOUT GND1	AIN1-	AIN5-	DIO3	DIO11	DIO19	SYNC1_B
11	Shield	Shield	Shield	Shield	Shield	Shield	Shield	Shield
12	A1	AOUT0	AIN0+	AIN4+	DIO0	DIO8	DIO16	SYNC0_A
13	A2	AOUT GND0	AIN0-	AIN4-	DIO1	DIO9	DIO17	SYNC0_B
14	Shield	Shield	Shield	Shield	Shield	Shield	Shield	Shield

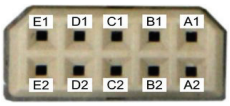
Table 12: Pin assignment of the COMBICON connector

8. Connector Assignment of AMC-ADIO24-HD50

8.1 Trigger Ports

Device connector: 10-pin har-link® female connector, P2
For the position of the connector in the front panel see page 15.

Pin Position:



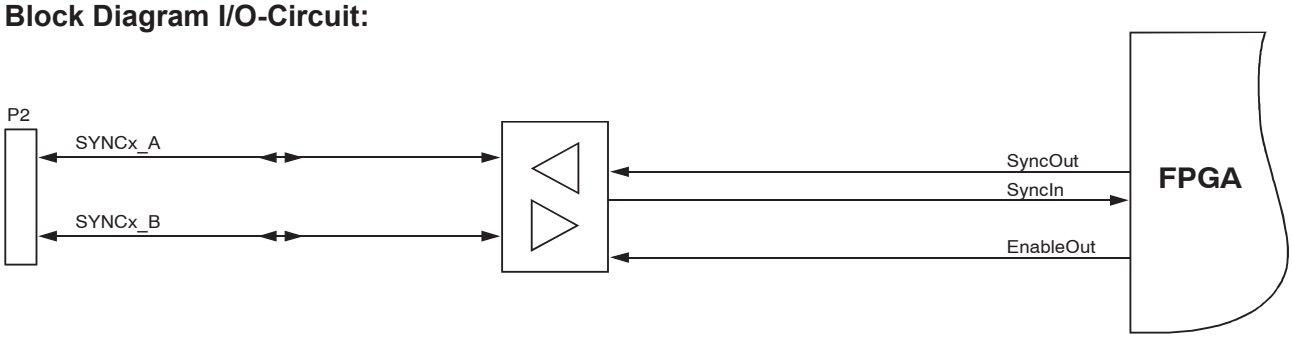
Pin Assignment:

SYNC0 ... SYNC3 (P2)					
Pin	**E1**	**D1**	**C1**	**B1**	**A1**
Signal	SYNC3_A	SYNC2_A	-	SYNC1_A	SYNC0_A
Pin	**E2**	**D2**	**C2**	**B2**	**A2**
Signal	SYNC3_B	SYNC2_B	GND	SYNC1_B	SYNC0_B

Signal Description:

SYNCx_A,
SYNCx_B signal lines of the RS-485 ports (trigger/sync) of channel x (x = 0 ... 3)
GND reference potential of trigger/sync I/Os

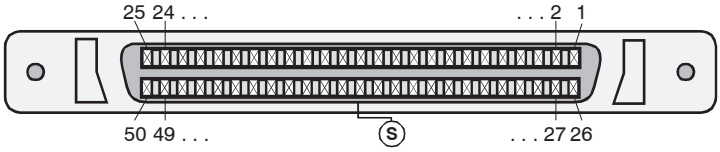
Block Diagram I/O-Circuit:



8.2 ADIO via 50-pin har.mik connector

Device connector: 50-pin har.mik® female connector, P1, Harting order no.: 60 01 050 5140
For the position of the connector in the front panel see page 15.

Pin Position:



Pin Assignment:

Signal	Pin	Pin	Signal
AOUT0-	1	26	AOUT1-
AOUT0+	2	27	AOUT1+
AOUTGND0	3	28	AOUTGND1
AOUT0	4	29	AOUT1
AIN0+	5	30	AIN0-
AIN1+	6	31	AIN1-
AIN2+	7	32	AIN2-
AIN3+	8	33	AIN3-
AIN4+	9	34	AIN4-
AIN5+	10	35	AIN5-
AIN6+	11	36	AIN6-
AIN7+	12	37	AIN7-
DIO0	13	38	DIO1
DIO2	14	39	DIO3
DIO4	15	40	DIO5
DIO6	16	41	DIO7
DIO8	17	42	DIO9
DIO10	18	43	DIO11
DIO12	19	44	DIO13
DIO14	20	45	DIO15
DIO16	21	46	DIO17
DIO18	22	47	DIO19
DIO20	23	48	DIO21
DIO22	24	49	DIO23
GND	25	50	GND
		S	Shield

Signal Description:

AOUTx*+/-...	analog output signal lines (x = 0,1)
AOUTGNDx+/-...	reference potential of analog outputs
AINy+/-...	analog input signals (y = 0 ... 7)
DIOz...	digital IO-signals (z = 0 ... 23)
GND...	reference potential
Shield...	case shield, connected with the shield potential of the µTCA system.

9. AMC-ADIO24-HD50-Adapter

INFORMATION
The AMC-ADIO24-HD-Adapter (order No.: U.1001.09) is intended for the usage with the AMC-ADIO24-HD50 (U.1001.02).

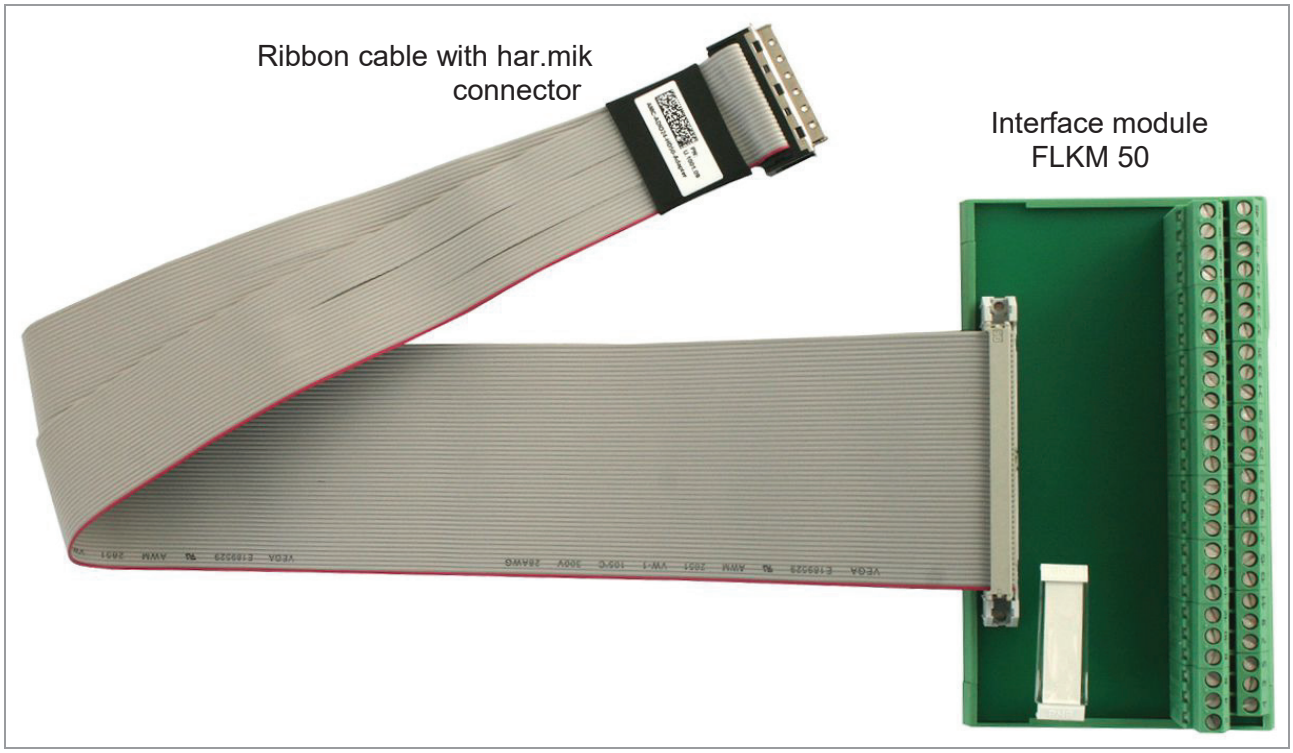


Figure 11: AMC-ADIO24-HD50-Adapter

The AMC-ADIO24-HD50-Adapter is designed to be connected to the har.mik connector of the AMC-ADIO24-HD50 (see page 100).
The adapter comes with a flat-ribbon cable with the har-mik mating connector (male) and the interface module FLKM 50.
The interface module FLKM is equipped with a flat-ribbon cable connector and screw connections for the 50 channels. The signals are passed through (1-1 connection).

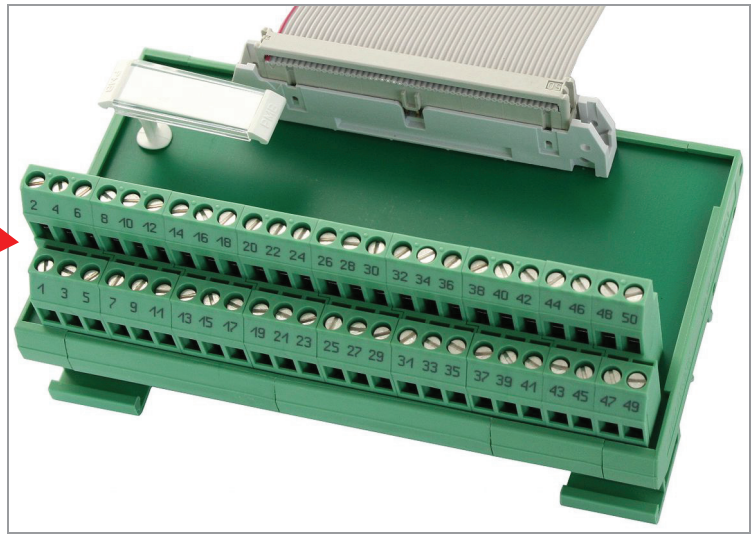
9.1 Technical Data AMC-ADIO24-HD50-Adapter

Connectors	AMC-ADIO24- HD50 side: 50-pin har.mik® connector, male, at the ribbon cable Process side: Phoenix Contact connector with screw connections, 50 positions; Stripping length: 8 mm, Screw thread: M3, Conductor cross section solid: 0.2 mm² ... 4 mm², Conductor cross section flexible: 0.2 mm² ... 2.5 mm², Conductor cross section AWG: 24 ... 12
Temperature range	-20 °C ... +50 °C ambient temperature
Humidity	max. 90%, non-condensing
Dimensions	Module only: 135 mm x 77 mm x 58.5 mm (width, height, depth) Cable length: 50 mm
IP Class	IP20
UL-Flame Rating	UL94 V-0
Housing	Plastic housing of FLKM50 for carrier rail mounting NS 32 or NS35/7,5 DIN EN 60715

Table 13: General data of the AMC-ADIO24-HD50-Adapter

9.2 Connector Assignment

Connector with screw-connections



Pin Assignment:

Signal	Pin	Pin	Signal
AOUT0-	1	2	AOUT0+
AOUTGND0	3	4	AOUT0
AIN0+	5	6	AIN1+
AIN2+	7	8	AIN3+
AIN4+	9	10	AIN5+
AIN6+	11	12	AIN7+
DIO0	13	14	DIO2
DIO4	15	16	DIO6
DIO8	17	18	DIO10
DIO12	19	20	DIO14
DIO16	21	22	DIO18
DIO20	23	24	DIO22
GND	25	26	AOUT1-
AOUT1+	27	28	AOUTGND1
AOUT1	29	30	AIN0-
AIN1-	31	32	AIN2-
AIN3-	33	34	AIN4-
AIN5-	35	36	AIN6-
AIN7-	37	38	DIO1
DIO3	39	40	DIO5
DIO7	41	42	DIO9
DIO11	43	44	DIO13
DIO15	45	46	DIO17
DIO19	47	48	DIO21
DIO23	49	50	GND

Signal Description:

AOUTx*+/-...	analog output signal lines (x = 0,1)
AOUTGNDx+/-...	reference potential of analog outputs
AINy+/-...	analog input signals (y = 0 ... 7)
DIOz...	digital IO-signals (z = 0 ... 23)
GND...	reference potential

10. Abbreviations

0x1234	hexadecimal value 1234
acc.	access
ADC	analog/digital converter
addr.	address
attr.	attribute
bwl	byte + word + long access possible
DAC	digital/analog converter
DDFS	direct digital frequency synthesizer
DMA	direct memory access
DREQ	DMA request
freq.	frequency
Hex	hexadecimal
I/O	input/output
in	input
Irq	interrupt
KSPS	kilo samples per second
l	long
MSPS	mega samples per second
n/a	not applicable
out	output
R	read access
RA	read address
reg. no.	register number
RO	read only access
SPI	serial peripheral interface (bus)
sync	synchronisation
TRP	timing routing pool (see page 33)
TS	timestamp
W	write access
WA	write address
wl	word + long access possible
WO	write only access

11. Declaration of Conformity of AMC-ADIO24

EU-KONFORMITÄTSERKLÄRUNG
EU DECLARATION OF CONFORMITY

Adresse / *Address*
esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

esd erklärt, dass das Produkt
esd declares, that the product
AMC-ADIO24

Typ, Modell, Artikel-Nr.
Type, Model, Article No.
U.1001.01

die Anforderungen der Normen
fulfills the requirements of the standards
EN 61000-6-2:2005,
EN 61000-6-4:2007/A1:2011

gemäß folgendem Prüfbericht erfüllt.
according to test certificate.
H-K00-0467-12

Das Produkt entspricht damit der EU-Richtlinie „EMV"
Therefore the product conforms to the EU Directive 'EMC'
2014/30/EU

Das Produkt entspricht der EU-Richtlinie „RoHS"
The product conforms to the EU Directive 'RoHS'
2011/65/EU

Diese Erklärung verliert ihre Gültigkeit, wenn das Produkt nicht den Herstellerunterlagen entsprechend eingesetzt und betrieben wird, oder das Produkt abweichend modifiziert wird.
This declaration loses its validity if the product is not used or run according to the manufacturer's documentation or if non-compliant modifications are made.

Name / *Name* T. Ramm
Funktion / *Title* CE-Koordinator / *CE Coordinator*
Datum / *Date* Hannover, 2018-01-05

Rechtsgültige Unterschrift / *authorized signature*

I:\Texte\Doku\MANUALS\AMC\AMC-ADIO24\Konformitaet\AMC-ADIO24_Konformitaetserklaerung_2018-01-05.odt

12. Order Information

Type	Properties	Order No.
AMC-ADIO24	8x analog input, 2x analog output, 24x digital I/O and 4x trigger via 7x 10-pin Harting har-link connectors	U.1001.01
AMC-ADIO24-HD50	8x analog input, 2x analog output and 24x digital I/O via 50-pin Harting har.mik connector, 4x trigger via 1x 10-pin Harting har-link connector	U.1001.02
Accessories for AMC-ADIO24:		
AMC-ADIO-Adapter	DIN EN rail module for conversion from 10-pin har-link® interface female connector to a 14-pin MC 1,5 COMBICON female connector	U.1001.10
har-link® cable	Adapter cable, 10-pin har-link® male connector to 10-pin har-link® male connector, length 0.5 m length 1.0 m length 2.0 m	 U.1001.11 U.1001.12 U.1001.13
Accessories for AMC-ADIO24-HD50:		
AMC-ADIO24-HD50-Adapter	Adapter from SCSI series connector (50-pin har.mik connector), via ribbon cable (0,5m) to transfer / interface modules (50-pole, screw-connection, mounted on standard DIN rail)	U.1001.09

Table 14: Order information

PDF Manuals
Manuals are available in English and usually in German as well. For availability of English manuals see table below.
Please download the manuals as PDF documents from our esd website www.esd.eu for free.

Manuals		Order No.
AMC-ADIO24/-HD50-ME	Hardware manual in English	U.1001.21

Table 15: Available manuals

Printed Manuals
If you need a printout of the manual additionally, please contact our sales team: sales@esd.eu for a quotation. Printed manuals may be ordered for a fee.